Lập trình hệ thống nhúng sử dụng vi điều khiển MSP430 (Embedded System I)

Ts. Lê Mạnh Hải

Khoa CNTT,

ĐH Công nghệ TP HCM

11/2013

Mở đầu

I Mục đích môn học:

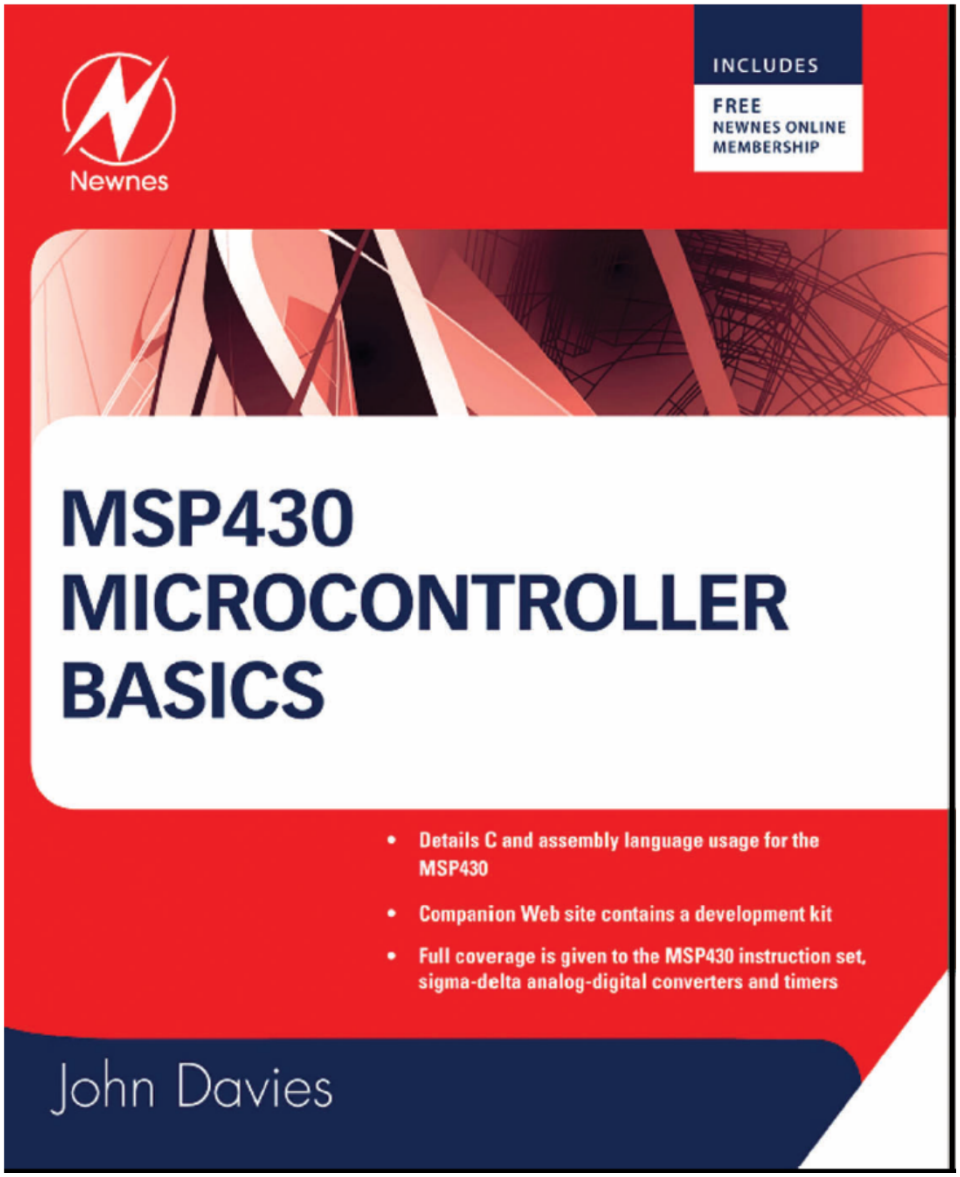
- Cung cấp kiến thức về lập trình vi điều khiển TI MSP430.

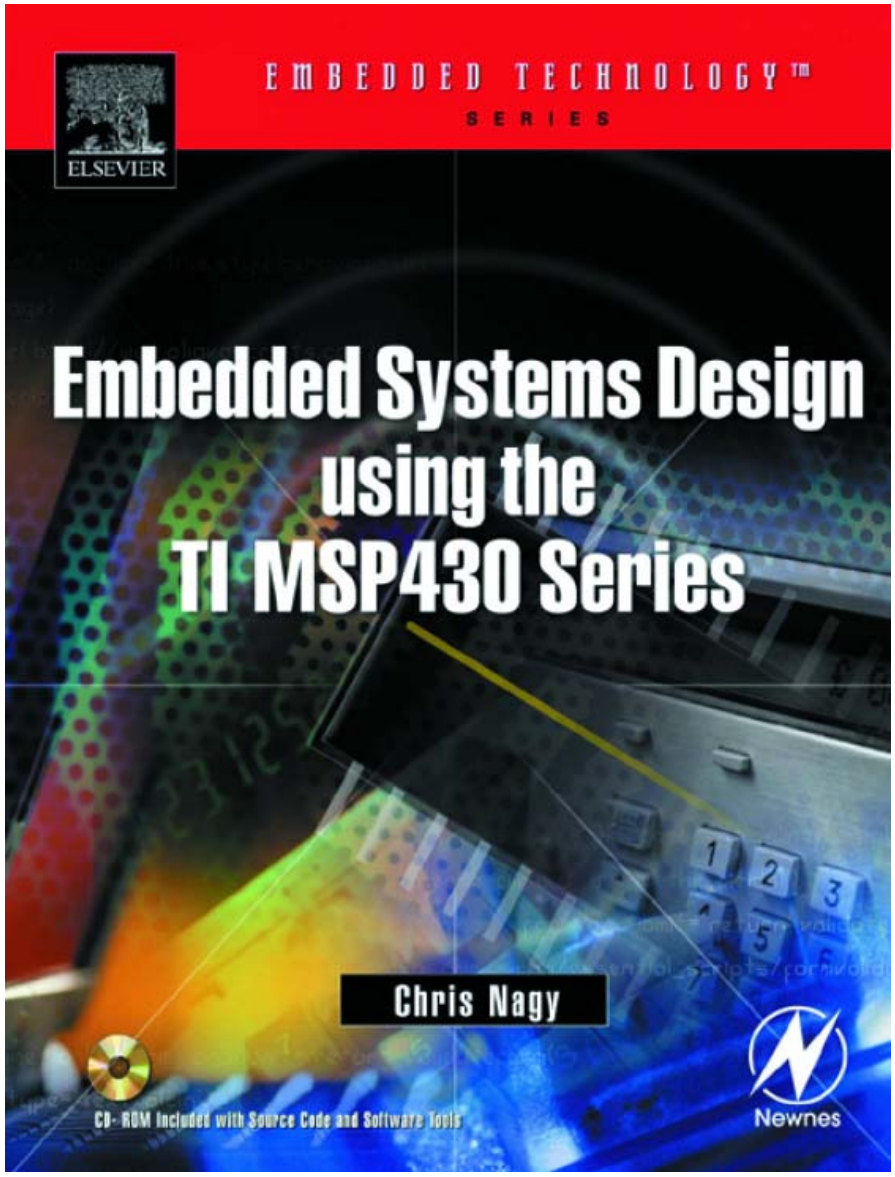
II. Thời gian:

- 30 tiết lý thuyết (2 tín chỉ) + 30 tiết thực hành (1 tín chỉ)

III Giáo trình và tài liệu tham khảo

- ***MSP430 Microcontroller Basics***. John H. Davies. Elsevier. 2008 (685 trang)
- ***Embedded Systems Design using the TI MSP430 Series.*** Chris Nagy. Elsevier. 2003 (296trang)
- Introduction to Embedded Systems - A Cyber-Physical Systems Approach, E. A. Lee and S. A. Seshia. http://LeeSeshia.org. 2011


Newnes
INCLUDES
FREE
NEWNES ONLINE
MEMBERSHIP
MSP430
MICROCONTROLLER
BASICS
Details C and assembly language usage for the MSP430
Companion Web site contains a development kit
Full coverage is given to the MSP430 instruction set, sigma-delta analog-digital converters and timers
John Davies


ELSEVIER
EMBEDDED TECHNOLOGY™
SERIES
Embedded Systems Design
using the
TI MSP430 Series
Chris Nagy
CD- ROM Included with Source Code and Software Tools
Newnes

IV. Đánh giá:

- Thi kết thúc môn: Bài tự luận với 3 câu hỏi.

V. Giáo viên:

- Ts. Lê Mạnh Hải. Tel: 0985399000.
- Không gọi điện thoại để hỏi hay xin điểm, email: hailemanh@yahoo.com, **lm.hai@hutech.edu.vn**
- Website: giangvien.hutech.edu.vn
- GV thực hành: Nguyễn Ngọc Đức. 0978629557

Nội dung chi tiết

Chương 1: Các hệ thống nhúng và vi điều khiển MSP430

Chương 2: Phát triển ứng dụng nhúng.

Chương 3: Các hàm và ngắt

Chương 4: Nhập/xuất

Chương 5: Bộ định thời

Chương 6: ADC

Chương 7: Kết nối

Chương 1: Các hệ thống nhúng và vi điều khiển MSP430

Sau khi học bài này, sinh viên sẽ nắm được

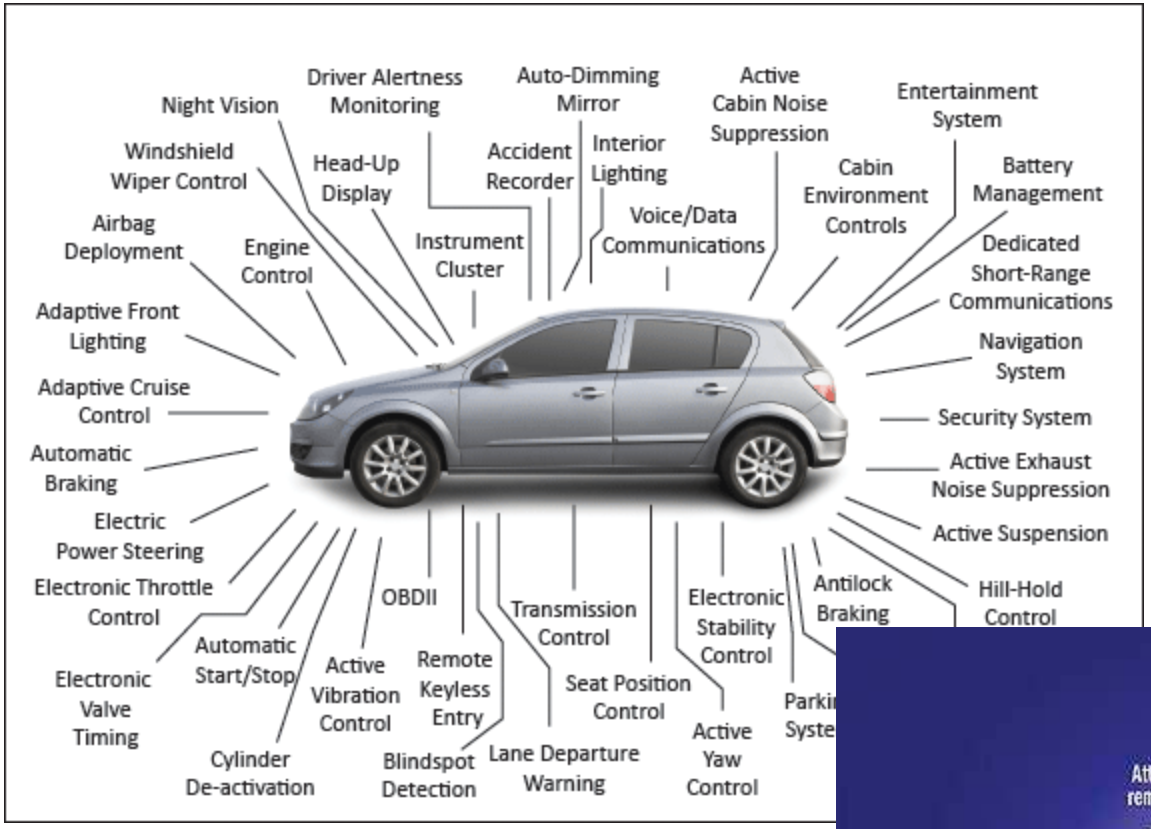
1. **Hệ thống nhúng là gì?**
2. **Các hướng phát triển hệ thống nhúng**
3. **Cấu trúc điển hình một vi điều khiển**
4. **Cấu trúc vi điều khiển MSP430G2553**

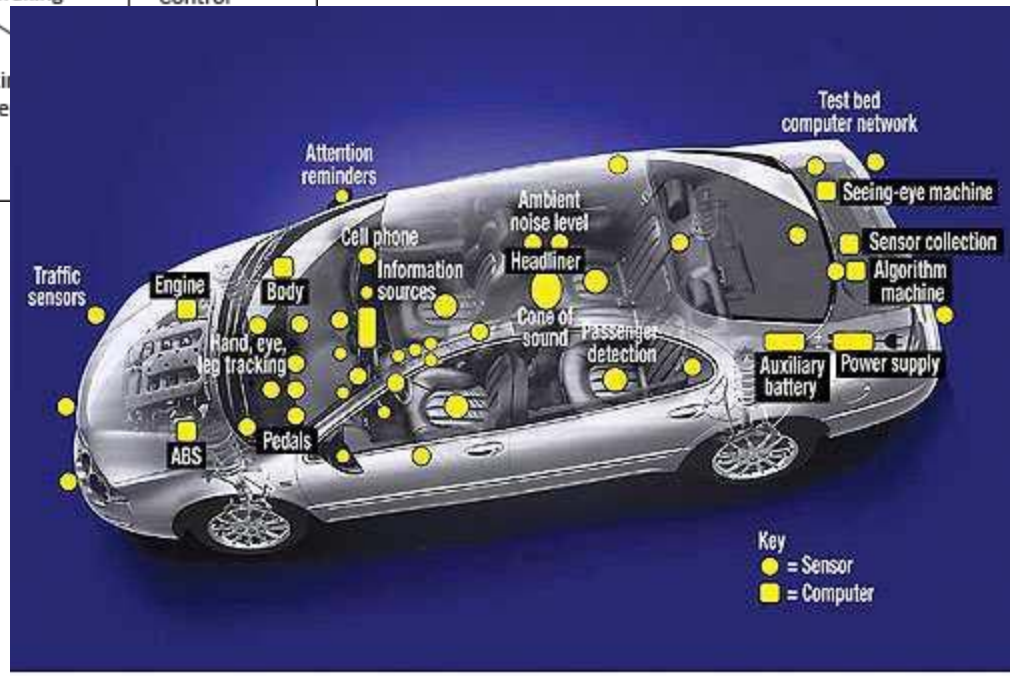
Hệ thống nhúng là gì?

- Theo vi.wikipedia.org: Hệ thống nhúng (Embedded system) là một thuật ngữ để chỉ một hệ thống có khả năng tự trị (**máy tính**) được **nhúng vào trong** một môi trường hay **một hệ thống mẹ**.
- Đó là các hệ thống tích hợp cả phần cứng và phần mềm phục vụ các bài toán chuyên dụng trong nhiều lĩnh vực công nghiệp, tự động hoá điều khiển, quan trắc và truyền tin. Đặc điểm của các hệ thống nhúng là hoạt động **ổn định** và có tính năng **tự động hoá cao**.

Hệ thống nhúng là gì?

- Máy giặt
- Xe hơi đời mới có trên 100 bộ xử lý
- Khoảng 99% chíp tính toán được ứng dụng trong các hệ thống nhúng





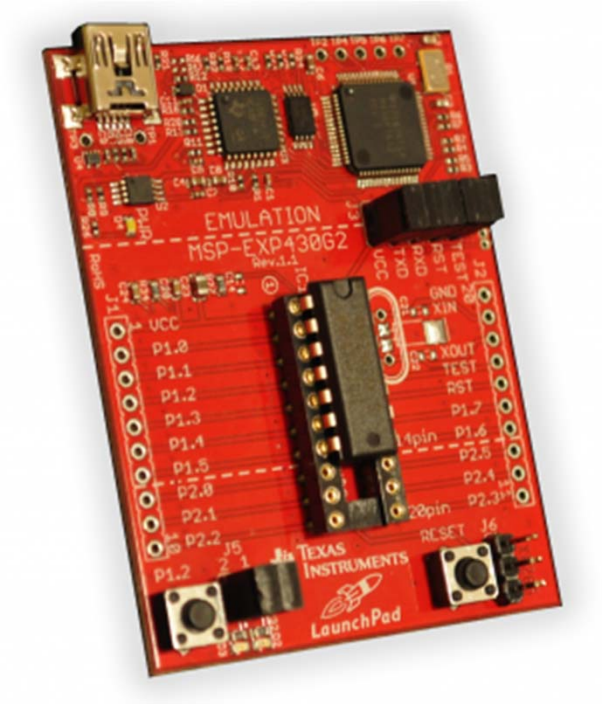
Automation of Cars- Embedded Systems Seminar For Electronics Students.

- Điện thoại di động thông minh (smartphone)
- TV

…

Bo Launchpad MSP430

MSP430G2543
MSP430G2553



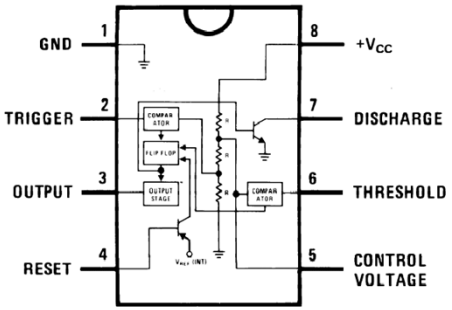
IAR Kickstart or
Code Composer
Studio Ver 5 (CCS)

MSP-EXP430G2 LaunchPad Experimenter Board

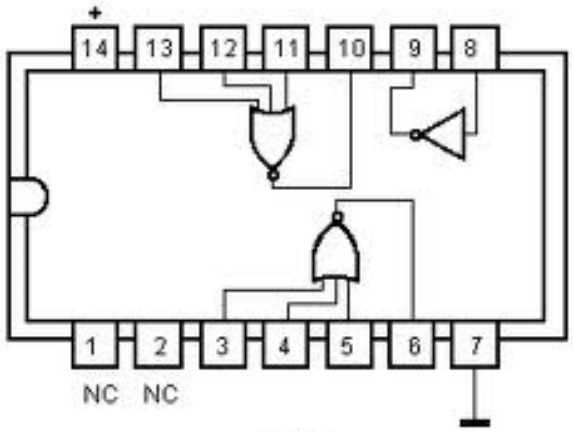
Các hướng phát triển hệ thống nhúng

- Một hệ thống điều khiển tương tự (trước năm 1970)
- Hệ thống máy tính số: Vi xử lý và vi điều khiển (1970 – nay)

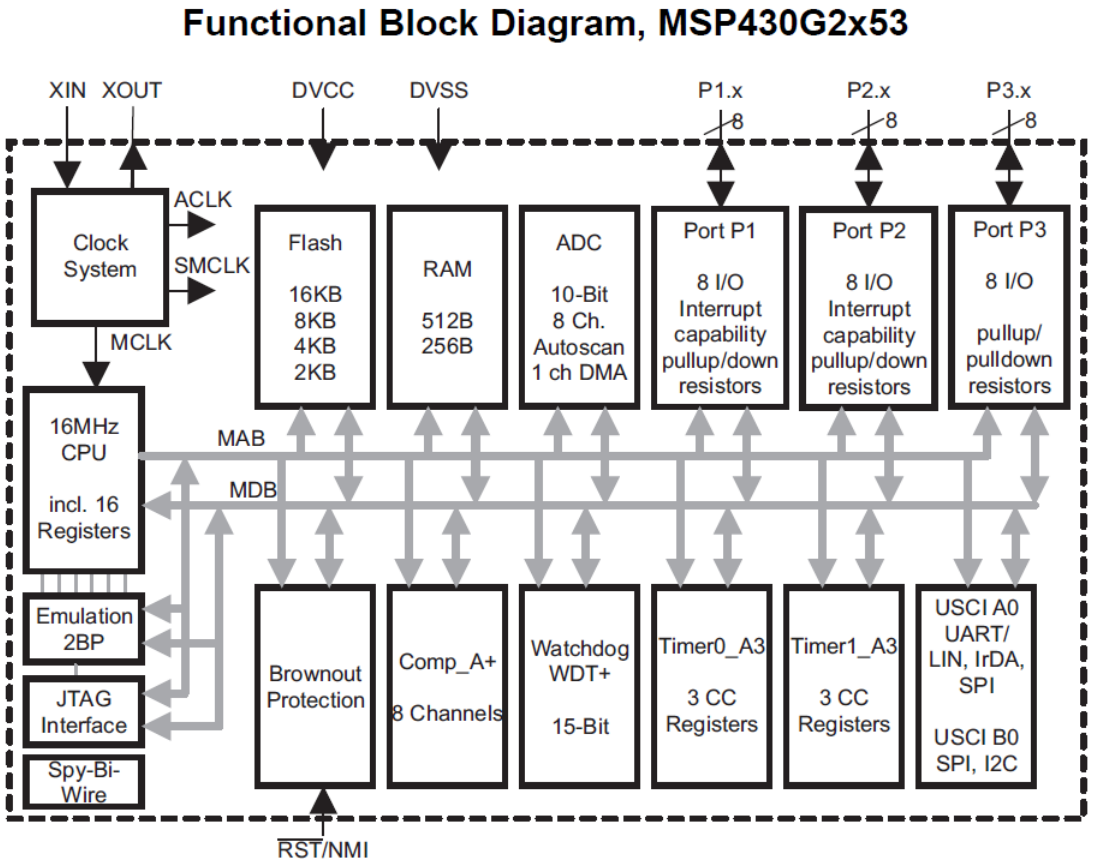
- ***Mạch số tích hợp thấp: transitor, IC 555***
- ***Mạch số tích hợp trung bình : CMOS 4000***
- ***Mạch số tích hợp cao: Vi điều khiển***



LM555 Timer



4000



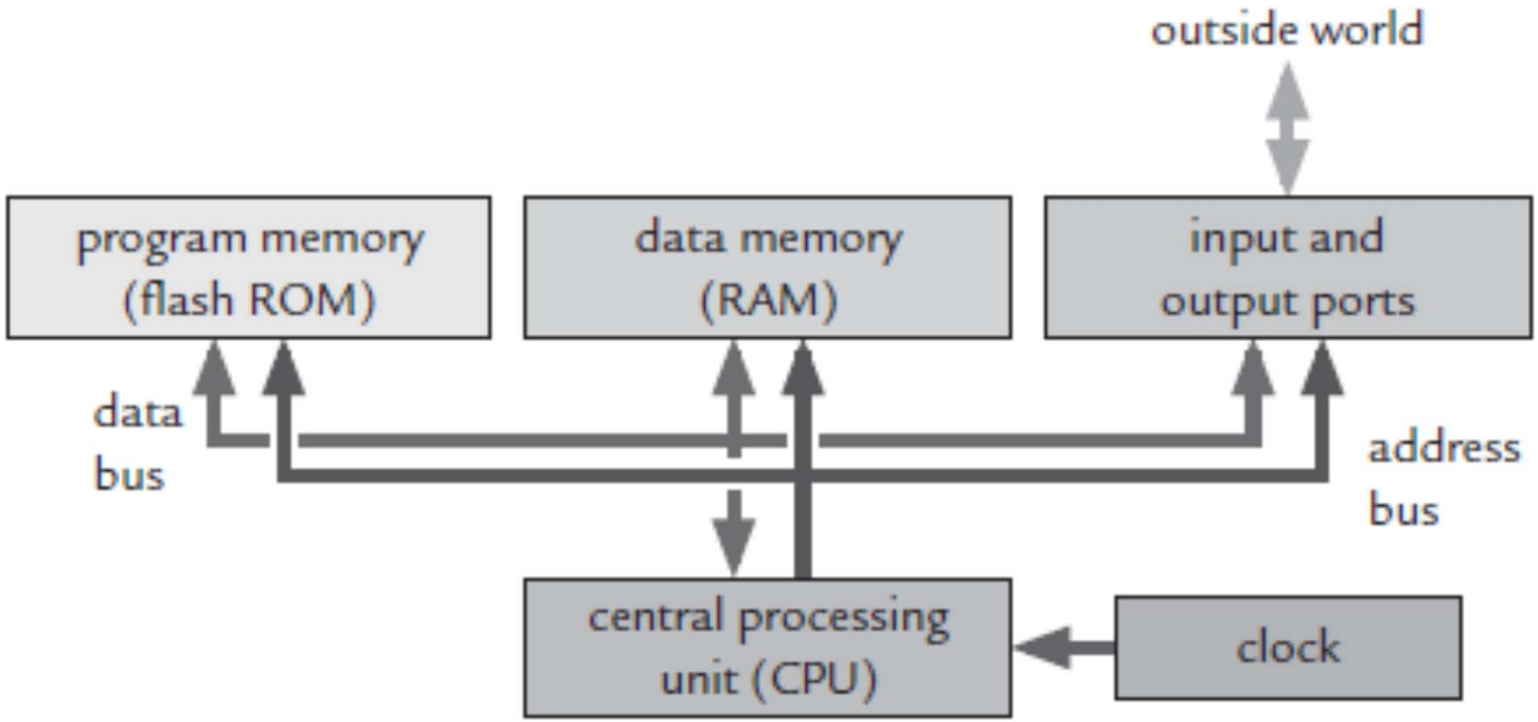
Các hướng ứng dụng

- **Application-specific integrated circuits (ASICs)**
  - Chíp (IC) thiết kế dành riêng cho một ứng dụng
- **Field-programmable gate arrays (FPGAs) and programmable logic devices (PLDs)**
  - Chíp thiết kế có thể lập trình thay đổi cấu tạo chức năng bằng cách tạo các mối liên kết giữa các cổng bên trong chíp. Có hàng triệu cổng trong một chíp.
- **Microcontrollers**
  - Có một số khối rất hay được sử dụng cùng với một khối xử lý trung tâm (CPU) .

Vi điều khiển nhỏ

- CPU xử lý 8 hoặc 16 bít
- Bộ nhớ 64 KB
- Tốc độ tối đa : 16Mhz
- Chức năng chính: điều khiển, không phải tính toán!
- http://www.diendanti.com

Cấu trúc chung của vi điều khiển



Vi điều khiển có 6 thành phần cơ bản sau:

1. Khối xử lý trung tâm (CPU) bao gồm:

- Khối tính toán số học/logic(ALU).
- Khối giải mã lệnh và các mạch hỗ trợ xử lý ngắt, tái khởi động
- Các thanh ghi bao gồm thanh ghi đếm chương trình PC, con trỏ ngăn xếp SP, thang ghi trạng thái (SR), thanh ghi tạo hằng số CG và 12 thanh ghi đa năng

2. Bộ nhớ chương trình: Là bộ nhớ không mất dữ liệu khi mất điện. Trước kia là ROM, nay sử dụng FLASH. Chíp MSP430G2553 chỉ có 16KB

3. Bộ nhớ dữ liệu: RAM truy xuất tùy ý nhưng dữ liệu bị xóa khi mất điện

- **Hiện đã có bộ nhớ dữ liệu không bị xóa khi mất điện**

4. Các cổng nhập/xuất: Kết nối với các hệ thống khác

5. Đường BUS dữ liệu và BUS địa chỉ: Để truyền dữ liệu và lệnh giữa các khối.

6. Khối xung nhịp: Tạo xung đồng bộ các khối

08 khối thường gặp khác:

Khối định thời (Timer): Đếm thời gian chính xác. Các vi điều khiển hiện nay có ít nhất 2 khối này.

Khối định thời cảnh báo: Là khối kiểm soát lỗi chương trình theo thời gian. Khối này sẽ tái khởi động chíp khi chương trình bị lỗi .

Khối giao tiếp tuần tự: Kết nối với các IC khác bằng cách truyền từng bít.

Khối nhớ dữ liệu không bay hơi: Lưu trữ dữ liệu ngay cả khi mất điện. Thường dùng để lưu cấu hình thiết bị như địa chỉ IP trong các ADSL router

Khối biến đổi tương tự - số : Cho phép chuyển đổi tín hiệu tương tự sang dạng số.

Khối biến đổi số -tương tự : Cho phép chuyển đổi tín hiệu tương tự sang dạng số, thường dùng để điều khiển động cơ bằng phương pháp xung số (PWM).

Đồng hồ thời gian thực: Lưu giữ giá trị năm tháng ngày.

Bộ nạp và chạy chương trình: Cho phép nạp chương trình từ máy tính vào bộ nhớ chương trình

Cấu trúc Harvard và von Neumann

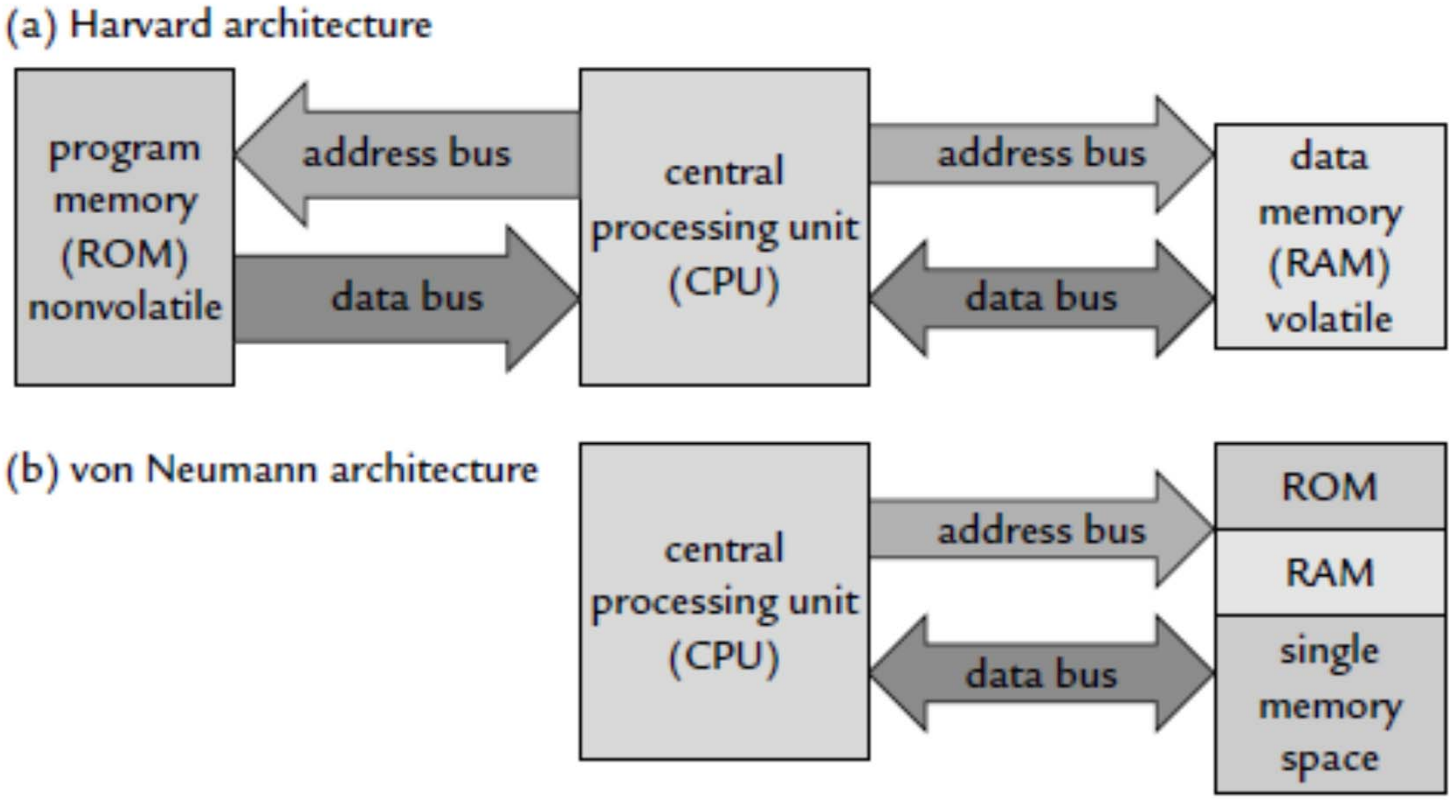


Figure 1.3: Harvard and von Neumann architectures for memory.

MSP 430 có cấu trúc von Neumann

Câu hỏi

- Hãy kể tên một vài thiết bị dân dụng là hệ thống nhúng?
- Các thành phần cơ bản của một vi điều khiển?
- Các khối hỗ trợ thường gặp ở một VĐK?
- Sự khác biệt giữa cấu trúc **Harvard và von Neumann** ?

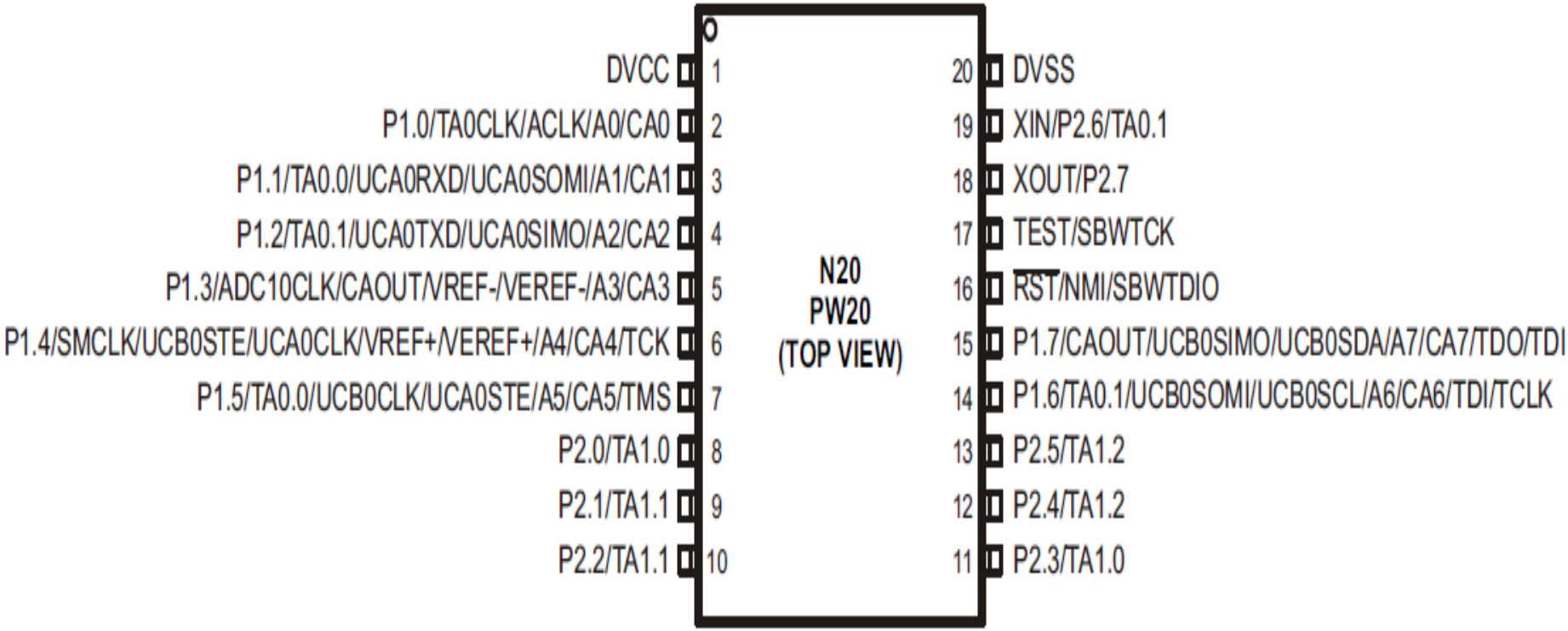
Bài 2: Cấu trúc vi điều khiển MSP430G2553

- **Các chân VĐK MSP430G2553**
- **Các khối chức năng**
- **Tổ chức bộ nhớ**
- **Khối xử lý trung tâm (CPU)**
- **Bộ tạo xung nhịp**
- **Ngắt và tái khởi động**
- **Các tài liệu chính thống**

Các chân VĐK MSP430G2553

- *Chân ra: 20 chân từ vỏ nhựa PDIP*
- Phần lớn các chân có nhiều chức năng.
  - Ví dụ chân số 3 có 5 chức năng
- Các ứng dụng tại mỗi thời điểm chỉ yêu cầu mỗi chân thực hiện một chức năng => không bị mâu thuẫn

Device Pinout, MSP430G2x13 and MSP430G2x53, 20-Pin Devices, TSSOP and PDIP



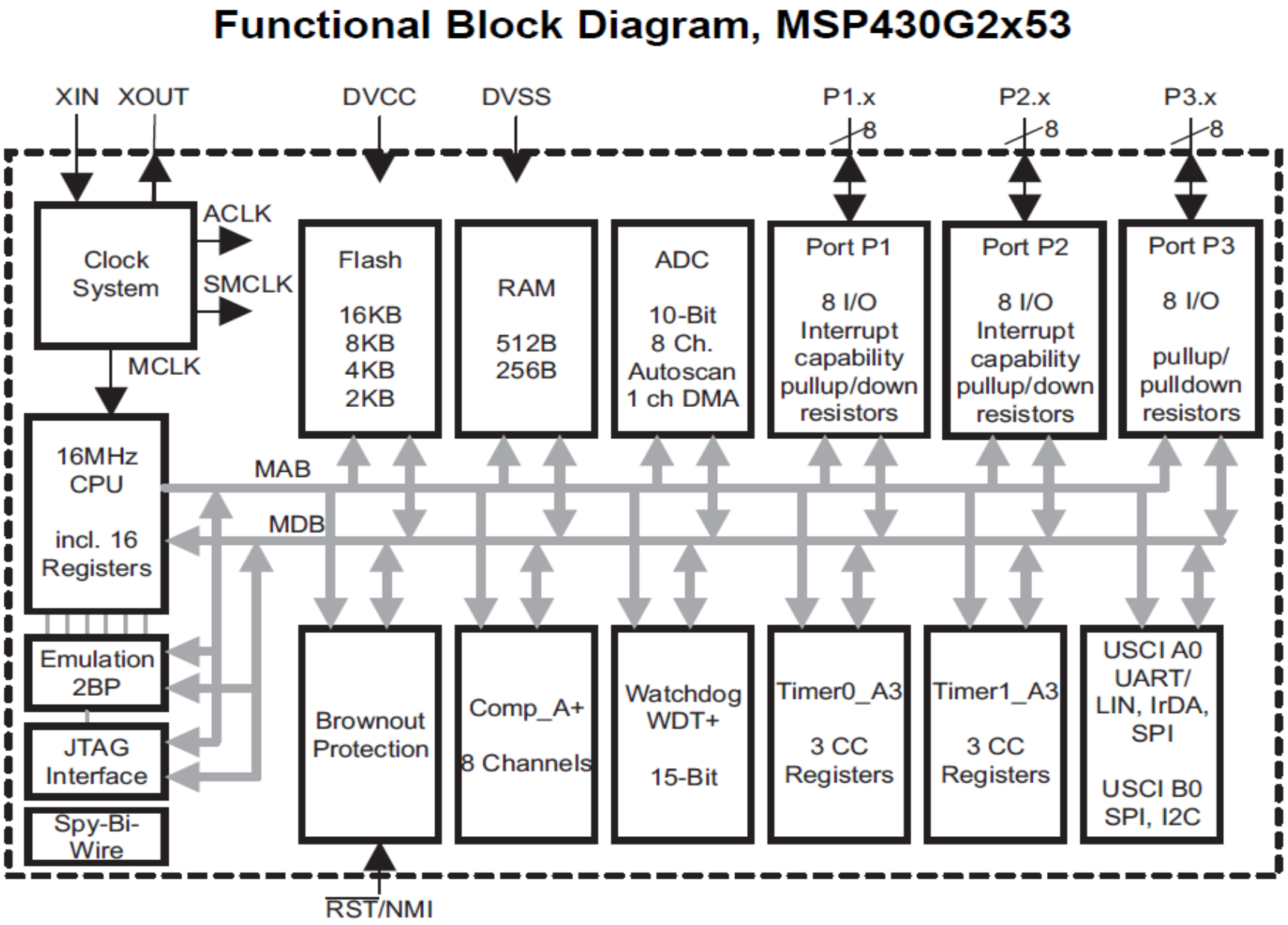
NOTE: ADC10 is available on MSP430G2x53 devices only.

NOTE: The pulldown resistors of port P3 should be enabled by setting P3REN.x = 1.

Mô tả các chân

- *V*CC(chân 1) *V*SS (chân 20) dùng để cấp nguồn 3,3V cho chíp. Nguồn có thể dao động trong khoảng 1,8V – 3,6 V
- P1.0–P1.7, P2.0, and P2.7 là 2 cổng nhập xuất số. Mỗi cổng 8 chân (8 bít), gọi tắt là P1 và P2.
- Các khối chức năng cũng sử dụng các chân này khi cần nhờ cấu hình thanh ghi chọn khối P1SEL và P2SEL.

Các khối chức năng MSP430G2553



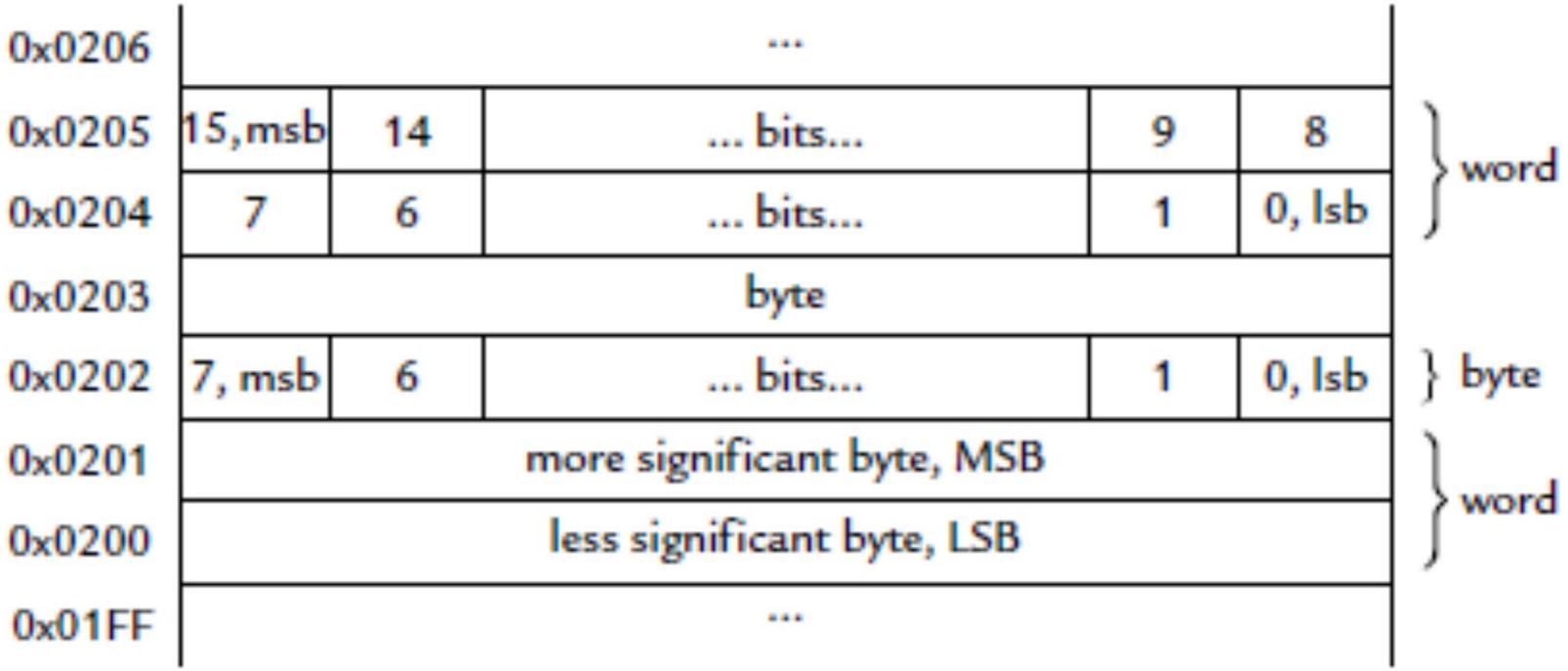
Mô tả các khối chức năng

- *Các khối cơ bản: CPU, Xung nhịp, Flash, RAM, Ports,và Bus.*
- Các khối bổ xung:
  - Khối nạp chương trình: JTAG : 4 dây và 2 dây
  - Khối biến đổi tương tự - số ADC.
  - Khối bảo vệ sụt áp (Brownout Protection)
  - Khối so sánh áp (Compare A+)
  - Khối Đồng hồ canh gác (WDT)
  - 2 khối định thời loại A (Timer0_A3 và Timer_A3)
  - 2 khối giao tiếp tuần tự (USCI A0 và USCI B0)

Bộ nhớ

- Bộ nhớ là các thanh ghi 8 bít, tổ chức thành các ô nhớ,
- Địa chỉ ô nhớ 16 bit từ 0x0000 tới 0xFFFF
- Bus dữ liệu 16 bit có thể truyền 16 bít hoặc 8 bít.

Bus địa chỉ và các ô nhớ



Thứ tự ô nhớ

- **Little-endian ordering:** Khi dữ liệu có trên 1 byte thì byte giá trị thấp nằm ở vị trí dưới, byte giá trị cao nằm ở bên trên trong bộ nhớ. **Dòng MSP430 có thứ tự này**.
- **Big-endian ordering:** Byte giá trị thấp nằm ở vị trí cao. Một số chíp của Motorola, Freescale HCS08 có cấu trúc này.

Tổ chức bộ nhớ

Tổ chức bộ nhớ của MSP430G2553 gồm các thành phần sau

- **Thanh ghi chức năng chuyên dụng**: Các thanh ghi của các khối có chức năng xác định trước. Ví dụ các thanh ghi PC, SP, SR, CG của CPU, thanh ghi P1REN, P1DIR của P1...
- **Các thanh ghi đa năng** của CPU và các thiết bị ngoại vi, như các thanh ghi R4-R15 của CPU, P1IN, P1OUT của P1.
  - Các thanh ghi 8 bít
  - Các thanh ghi 16 bít
- **Random access memory (RAM):** Các thanh ghi đặt trong khối RAM có địa chỉ từ 0x0200 và chỉ có 256/512 Bytes
- **Bootstrap loader :** Là phần bộ nhớ không bị xóa chứa chương trình kết nối máy tính qua cổng COM của TI

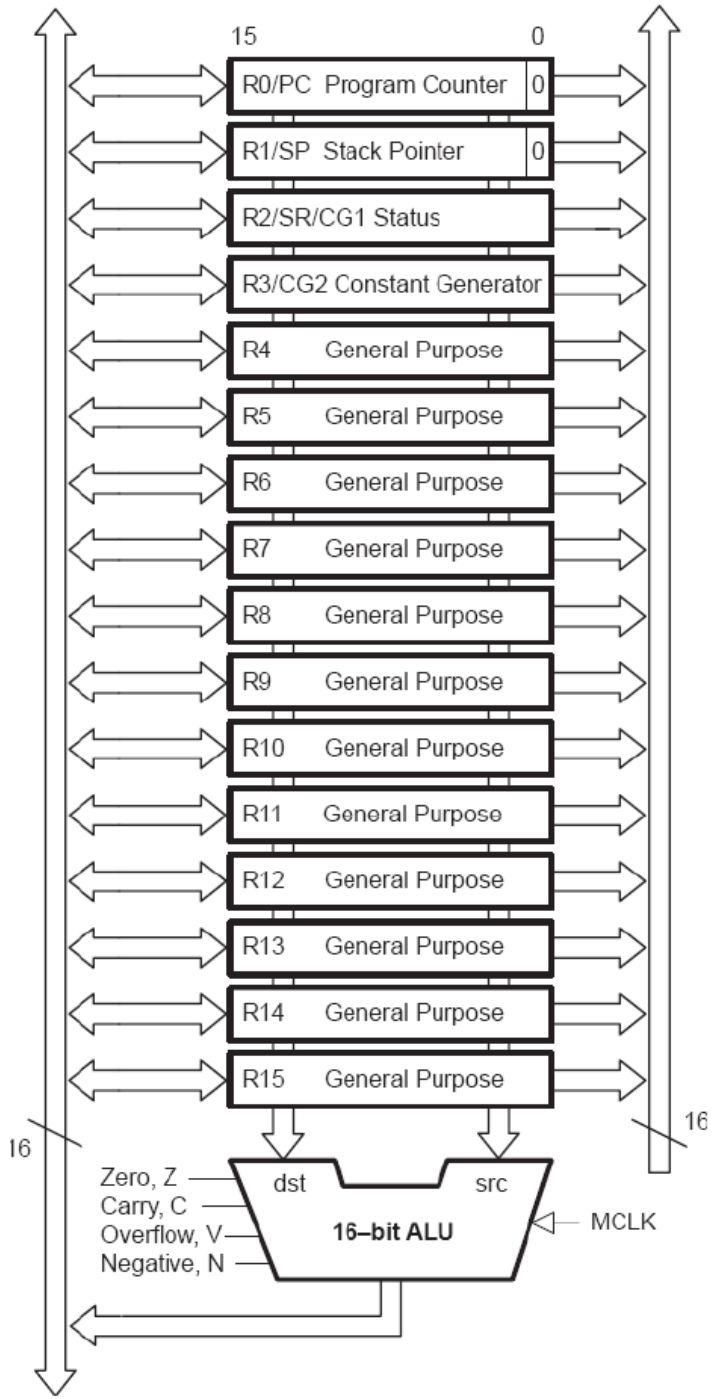
- **Bộ nhớ dữ liệu không bay hơi (Information memory):** Là 256 Byte flash, cho phép lưu các thông tin quan trọng và không bị mất khi mất điện.
- **Bộ nhớ chương trình (Code memory):** Là bộ nhớ chỉ đọc (ROM) và chứa chương trình được nạp từ máy tính xuống. Sau khi nạp và khởi động, chương trình này sẽ được đọc vào CPU để thực thi . Chíp hiện nay có từ 2KB-16KB.
- **Interrupt and reset vectors:** Là phần bộ nhớ chứa các địa chỉ của các hàm xử lý ngắt và tái khởi động.

Address	Type of memory
0xFFFF	interrupt and reset
0xFFC0	vector table
0xFFBF	flash code memory
0xF800	(lower boundary varies)
0xF7FF	
0x1100	
0x10FF	flash
0x1000	information memory
0x0FFF	*bootstrap loader*
0x0C00	*(not in F20xx)*
0x0BFF	
0x0280	
0x027F	RAM
0x0200	(upper boundary varies)
0x01FF	peripheral registers
0x0100	with word access
0x00FF	peripheral registers
0x0100	with byte access
0x000F	special function registers
0x0000	(byte access)

Khối xử lý trung tâm (CPU)

- Chức năng: Khối CPU thực thi các lệnh cất trong bộ nhớ chương trình. Các lệnh được đọc tuần tự và thực thi nếu không gặp các lệnh rẽ nhánh hoặc xử lý ngắt
- Cấu tạo: Gồm một khối tính toán ALU 16 bít, mạch giải mã lệnh và 16 thanh ghi.
- Tần số tối đa (cũng là tốc độ) do xung nhịp MCLK tạo là 16MHz.
-

CPU


15
0
R0/PC Program Counter
0
R1/SP Stack Pointer
0
R2/SR/CG1 Status
R3/CG2 Constant Generator
R4 General Purpose
R5 General Purpose
R6 General Purpose
R7 General Purpose
R8 General Purpose
R9 General Purpose
R10 General Purpose
R11 General Purpose
R12 General Purpose
R13 General Purpose
R14 General Purpose
R15 General Purpose
16
16
Zero, Z
Carry, C
Overflow, V
Negative, N
dst
src
16–bit ALU
MCLK

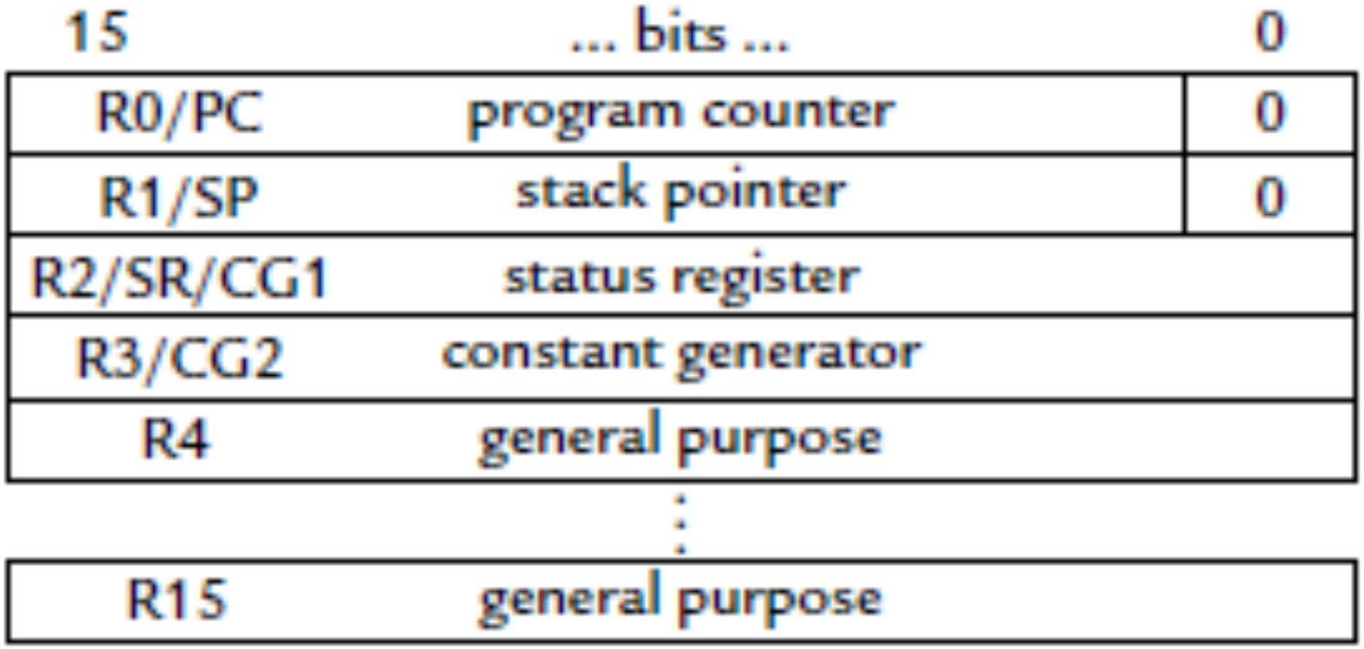


Figure 2.5: Registers in the CPU of the MSP430.

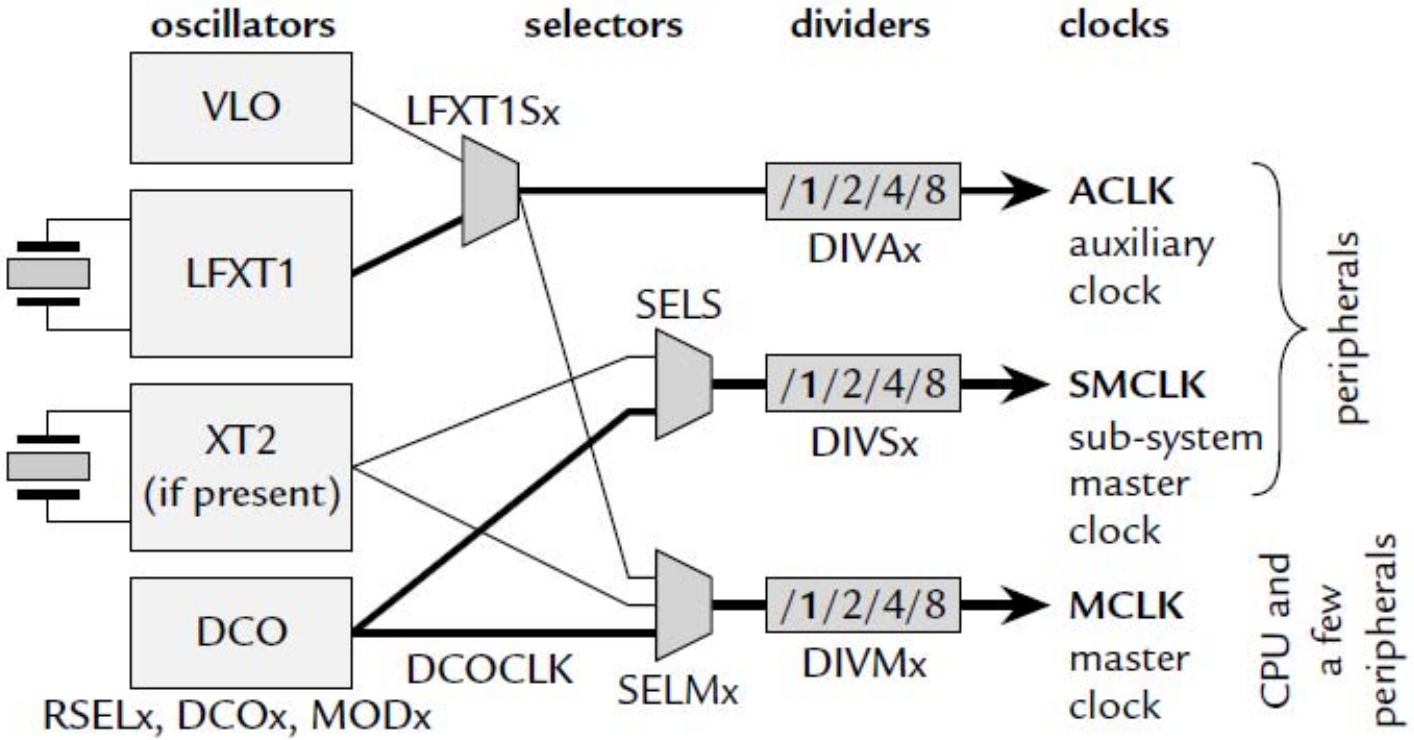
Các thanh ghi của CPU

- **Thanh ghi đếm chương trình** (Program counter – PC): Chứa địa chỉ lệnh kế tiếp cần thực hiện. PC tự động tăng 2 sau mỗi xung nhịp, ngoại trừ có lệnh rẽ nhánh hoặc gọi hàm
- **Con trỏ ngăn xếp** (Stack pointer- SP): Trỏ đến vùng nhớ RAM dùng làm ngăn xếp. Khi một hàm được gọi, PC và SR được cất vào ngăn xếp và khi thực hiện xong hàm, các giá trị này được trả lại PC và SR để tiếp tục oc6ng việc đang thực hiện dở
- **Thanh ghi trạng thái** (Status register – SR): Chứa các cờ trang thái. Các bít trạng thái hay được dùng là C, Z, N, và V. Ngoài ra có một số bít để tắt xung nhịp như CPUOFF (tắt MCLK)
- **Thanh ghi hằng số** (Constant generator) Dùng để tạo ra một số hằng số thường gặp
- **12 thanh ghi đa năng** : là các thanh ghi dùng để lưu thông tin trung gian. Các thanh ghi có tốc độ tru cập tương đương CPU

Bộ tạo xung nhịp

- Xung nhịp là thiết bị không thể thiếu của các hệ thống số
- Linh kiện thường dùng để tạo xung nhịp là thạch anh có thể tạo dao động khoảng vài MHz cho chíp (max = 16MHz)
- Tuy nhiên khi chạy tốc độ cao, chíp tiêu tốn nhiều điện. Các hệ thống di động cần tiết kiệm pin nên một số bộ phận chỉ chạy ở tần số thấp sẽ tiết kiệm và tăng thời gian sử dụng pin. Bộ tạo xung nhịp cần đa dạng hóa các chế độ hoạt động của VĐK.
- Nhiều ứng dụng nhúng phần lớn thới gian ở trạng thái ngủ (công suất thấp)-> Cần tắt xung nhịp khi có thể. Khi có sự kiện ngắt, CPU sẽ được cấp xung đồng bộ lại để hoạt động.
- .

Bộ tạo xung nhịp



Bộ tạo xung thấp tần bằng thạch anh LFXT1: Có thể tạo các xung nhịp từ vài chục KHz tới 1 MHz với độ chính xác cao.

Bộ tạo xung cao tần bằng thạch anh- XT2: Giống LFXT1 ngoại trừ tần số cao hơn (8-16MHz)

Bộ tạo xung tần số thấp VLO với độ chính xác thấp. Các VĐK có thể tạo xung nhịp tần số thấp mà không cần thạch anh. Dùng VLO nếu cần tiết kiệm năng lượng (kéo dài thời gian sử dụng giữa hai lần sạc pin)

Bộ tạo xung điều khiển kỹ thuật số (DCO): Tạo các xung nhịp tần số cao (8-10MHz) mà không dùng thạch anh. Các VĐK sử dụng DCO trong giai đoạn đầu khi khởi động (khoảng 1 giây) .

- **Kiểm soát các xung nhịp thông qua thanh ghi trạng thái (SR)**

- **CPUOFF** khóa MCLK, và dừng CPU cũng như các khối nào dùng MCLK để đồng bộ
- **SCG1** khóa SMCLK và các khối dùng MCLK để đồng bộ
- **SCG0** khóa DCO
- **OSCOFF** khóa VLO và LFXT1.

Các loại xung nhịp

- **Master clock, MCLK**, sử dụng cho CPU và một sô khối tốc độ cao (1-16MHz).
- **Subsystem master clock, SMCLK**, dùng cho các các khối tốc độ trung bình (1MHz)
- **Auxiliary clock, ACLK**, Dùng cho các khối tốc độ thấp (32KHz).

Khái niệm Ngắt và tái khởi động

- **Ngắt (Interrupts):** Là một sự kiện gây bởi phần cứng , mặc dù được cài đặt bằng phần mềm) và thường cần được xử lý ngay lập tức. Khi xảy ra một ngắt, VĐK dừng chương trình đang thực thi, cất các thông tin/ trang thái hiện tại (PC, SR) để chuyển sang chương trình xử lý ngắt (ISR) **ngay lập tức**. Sau khi xử lý ngắt xong, VĐK quay lại thực hiện tiếp công việc đang thực hiện trước. Như vậy chương trình xử lý ngắt được gọi bởi phần cứng chứ không phải phần mềm.
- **Tái khởi động (Resets):** Được tạo bởi phần cứng, thường được dùng khi có sự kiện bất thường ngu hại khiến VĐK không thể tiếp tục công việc. Thông thường đồng hồ canh gác sẽ tái khởi động VĐK sau một khoảng thời gian nhất định nếu không được can thiệp kịp thời. Tái khởi động giúp hệ thống ở vào trạng thái ổn định

Câu hỏi

- Liệt kê các thanh ghi CPU
- Các nguồn xung và các loại xung của VĐK TI MSP430G2553?
- Ngắt là gì, tại sao hệ thống nhúng cần cơ chế ngắt?

Bài tiếp

- ***Môi trường phát triển ứng dụng***
- ***Chương trình đơn giản đầu tiên***

Bài 3: Môi trường phát triển ứng ứng dụng

Sau khi học xong SV cần nắm được:

1. **Các công cụ hỗ trợ phát triển ứng dụng**
2. **Ngôn ngữ lập trình nhúng C**
3. **Truy cập và gỡ rối**
4. **Bo Launchpad MSP430 và các chương trình bật tắt đèn LED**

Các công cụ phát triển ứng dụng

- **Bộ soạn CT (Editor):** Cho phép viết CT và kiểm tra cú pháp tức thì
- **Biên dịch (compiler): Chương trình chuyển mã C sang mã máy, có khả năng dò lỗi**
- **Gắn địa chỉ (Linker):** Phối hợp các thư viện và các hàm, gán địa chỉ bộ nhớ (để khi nạp vào VĐK không bị nhầm)
- **Bộ mô phỏng (Stand-alone simulator): Chương trình** phỏng tạo hoạt động của VĐK, thay cho chíp thật.
- **Embedded emulator/debugger:** Thiết bị cho phép nạp từng lệnh từ máy tính xuống VĐK để gỡ rối
- **Bộ nạp trực tiếp (In-circuit emulator):** Thiết bị gỡ rối cho phép VĐK chay từng lệnh từ trên máy tính. Giá khoảng 1000$
- **Chương trình nạp (Flash programmer):** Là phần mềm miễn phí của TI dùng để nạp chương trình từ máy tính vào bộ nhớ chương trình của VĐK.

- **Các phần mềm thông dụng**
  - **IAR EmbeddedWorkbench. Phần mềm miễn phí cho SV.**

    http://www.iar.com
  - Code Composer Studio
  - **Phần mềm miễn phí cho SV : 16KB chương trình**

The C Programming Language

```
if ((P1IN & BIT3) == 0) // Kiểm tra chân số 3 Port1 có bằng 0?
{
P1OUT = 0x01; //cấp ra port 1 giá trị  0b00000001
}
else
{
P1OUT = 0x00; //cấp ra port 1 giá trị 0b00000000
}
```

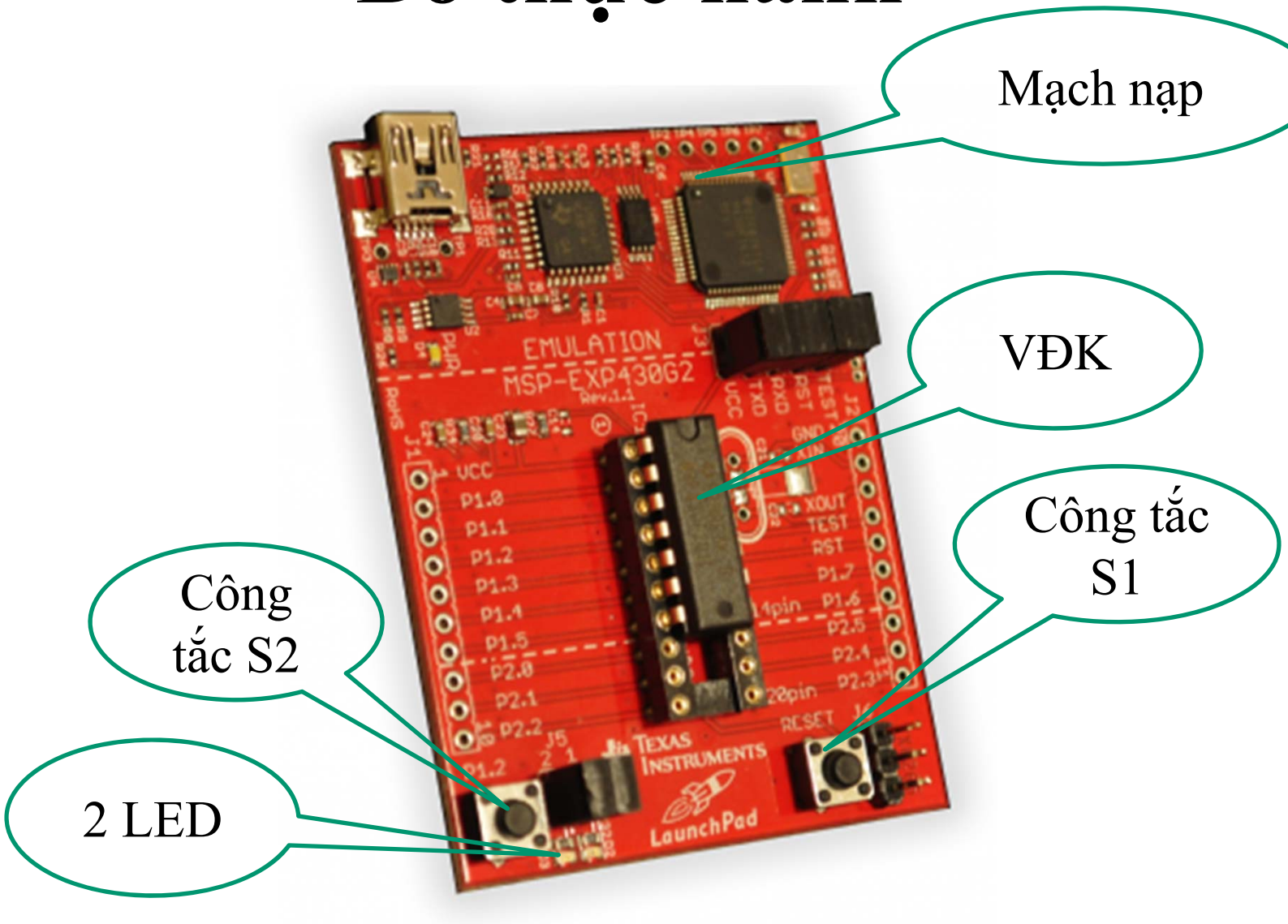
Hợp ngữ

Ví dụ:

mov.w #WDTPW|WDTHOLD ,& WDTCTL

Sinh viên sẽ học lập trình hợp ngữ nếu đã thành thạo lập trình bằng C

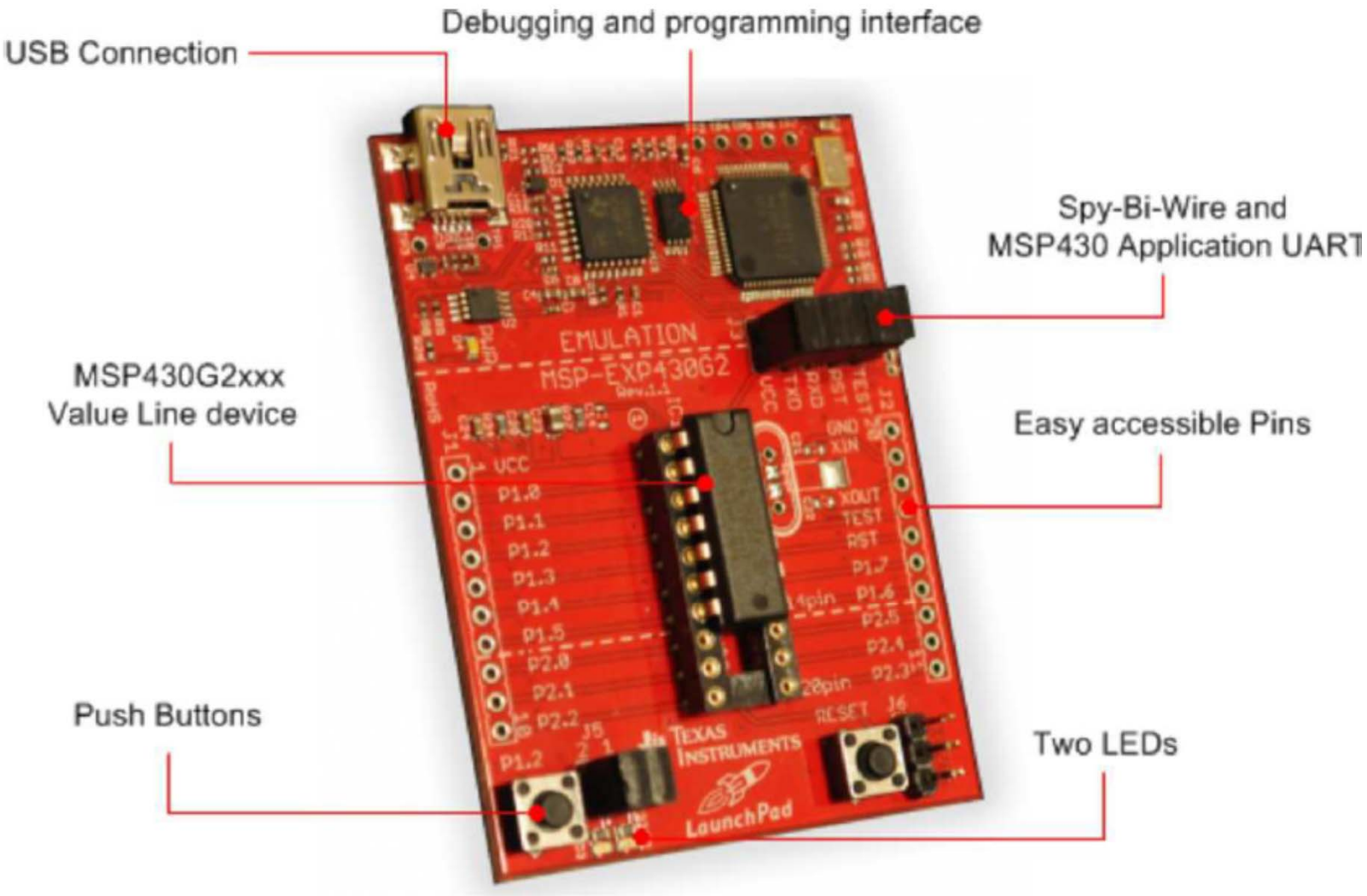
Bo thực hành

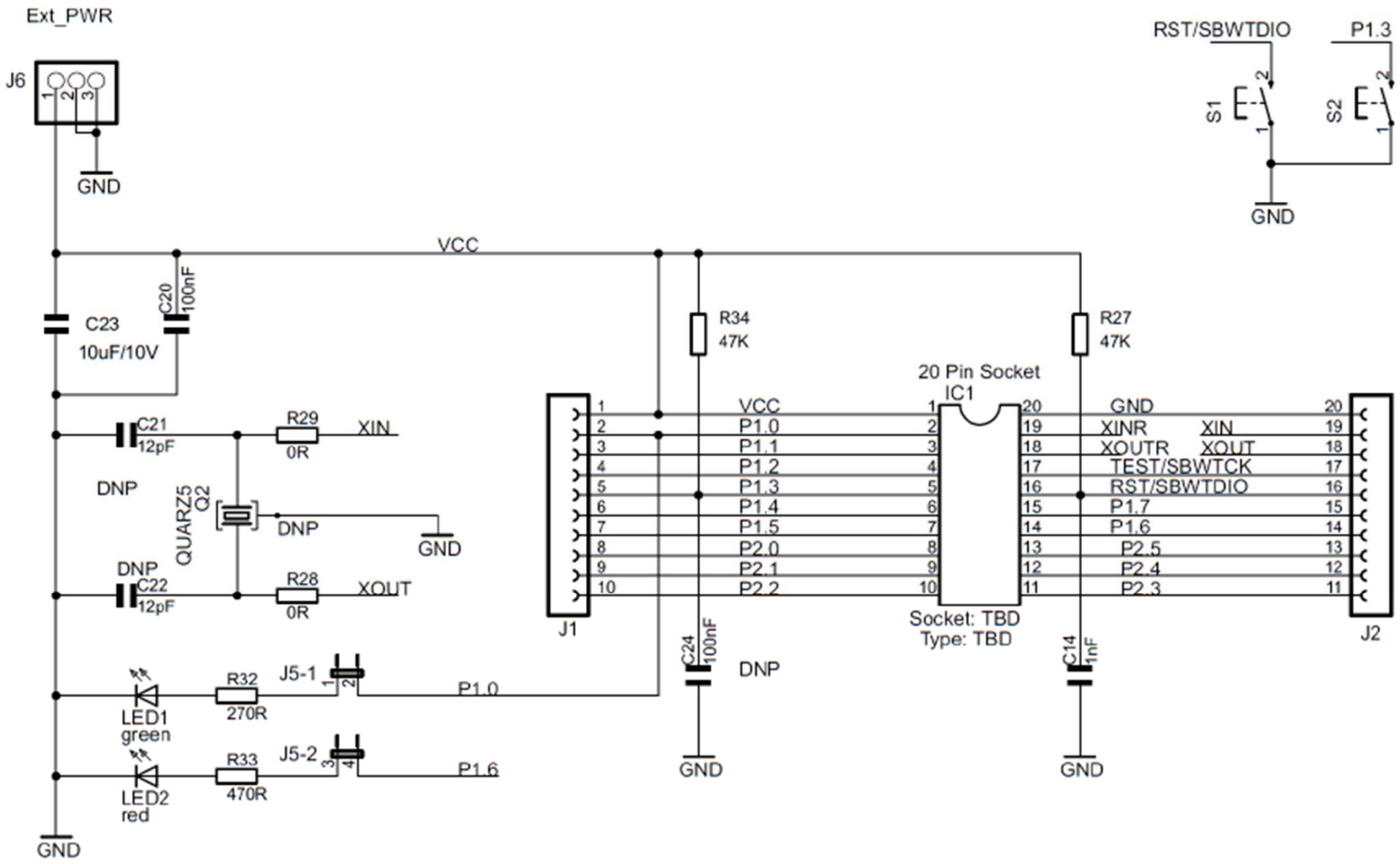


The TI MSP430G2553 Launchpad Board.

Các chương trình mở đầu

- **Bật /tắt các đèn LED**
- **Đọc trạng thái công tắc S2**
- **Tự động bật tắt đèn dùng hàm giữ chậm**
- **Automatic Control: Use of Subroutines**
- **Automatic Control: Flashing a Light by Polling Timer_A**
- **Header Files and Issues Brushed under the Carpet**


USB Connection
Debugging and programming interface
Spy-Bi-Wire and
MSP430 Application UART
MSP430G2xxx
Value Line device
Easy accessible Pins
Push Buttons
Two LEDs
EMULATION
MSP-EXP430G2
LaunchPad
TEXAS INSTRUMENTS


Ext_PWR
J6
GND
VCC
C23
10uF/10V
C20
100nF
C21
12pF
R29
0R
XIN
DNP
QUARZ5
Q2
DNP
GND
DNP
C22
12pF
R28
0R
XOUT
R32
270R
J5-1
P1.0
LED1
green
R33
470R
J5-2
P1.6
LED2
red
GND
R34
47K
R27
47K
20 Pin Socket
IC1
J1
VCC
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P2.0
P2.1
P2.2
GND
XINR
XOUTR
TEST/SBWTCK
RST/SBWTDIO
P1.7
P1.6
P2.5
P2.4
P2.3
XIN
XOUT
Socket: TBD
Type: TBD
J2
C24
100nF
DNP
GND
C14
1nF
GND
RST/SBWTDIO
P1.3
S1
S2
GND

Bật tắt các đèn LED

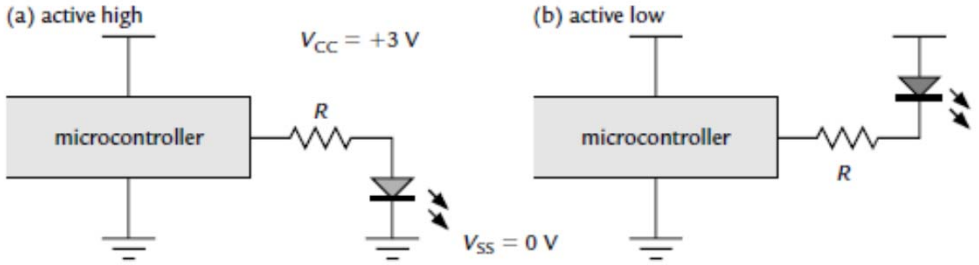
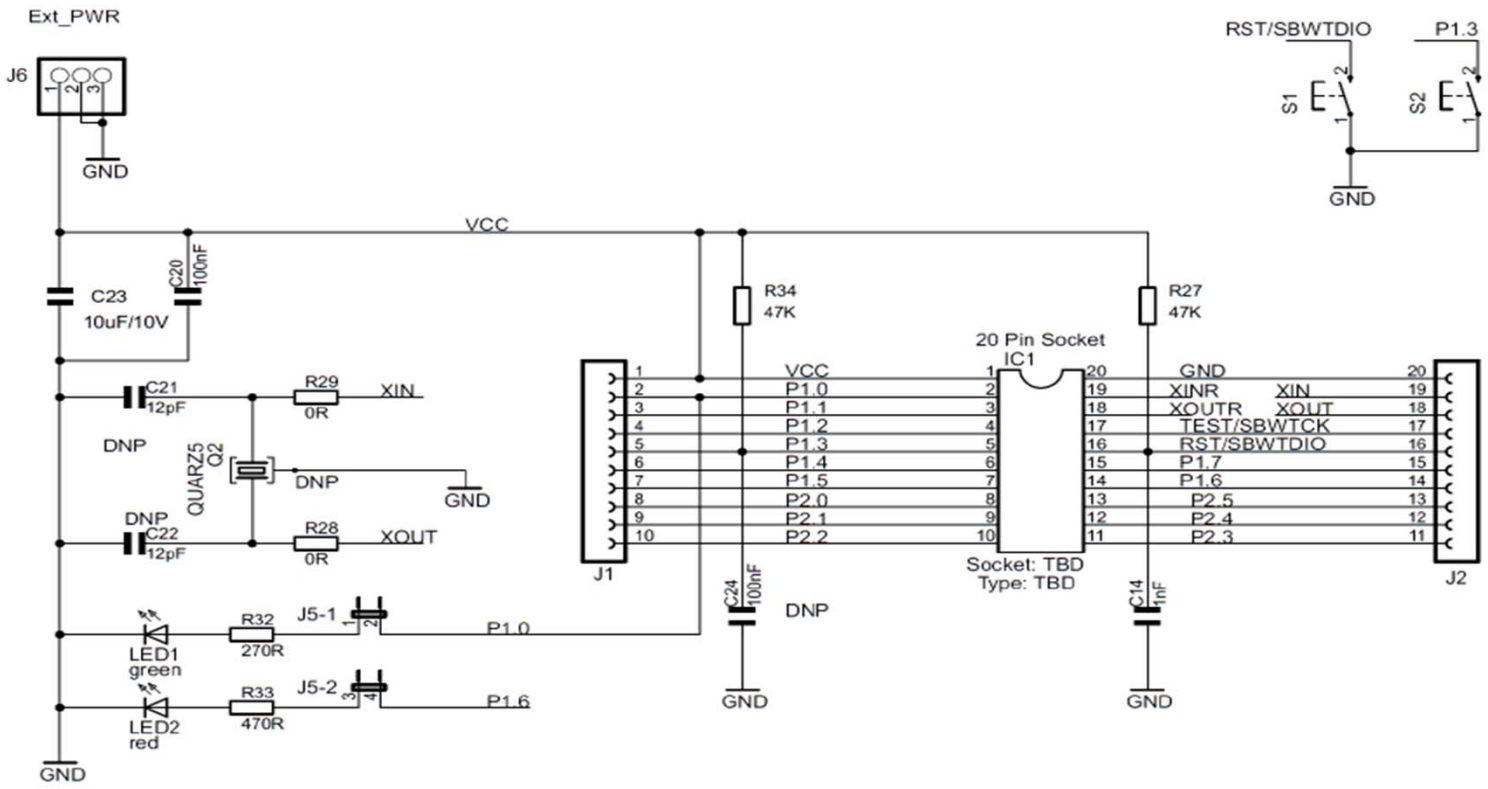


Figure 4.3: Connection of an LED to a pin of a microcontroller: (a) active high and (b) active low.

```
#include <msp430g2553.h> // Specific device
void main (void)
{
WDTCTL = WDTPW | WDTHOLD; // dừng  watchdog timer
P1DIR = 0x41; // đặt các chân P1.0 và P1.6 là OUTPUT:0b01000001
P1OUT = 0x41; // cả hai led cùng sáng
for (;;) { // Loop forever ...
} // ... doing nothing
}
```

Hãy sửa lại chuong trình để Led 1 sáng , led 2 tắt

Đọc trạng thái công tắc



Nếu công tắc S2 hở (mở) – điện áp chân P1.3 sẽ lên 1 (3,3v)

Nếu công tắc S2 đóng : điện áp trên P1.3 sẽ xuống 0 (0v)

```
#include <msp430g2553.h> // Specific device
#define LED1 BIT0
#define LED2 BIT6
#define S2 BIT3
void main (void)
{
WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer
P1DIR = LED1|LED2; // Set pin with LED1 and LED2 to output
P1OUT = LED1|LED2; // bat LED1 va LED2 sang
P1REN |= S2;    //su dung dien tro keo len/xuong, chi dung voi Launchpad v1.5
P1OUT |= S2;   //dien tro keo len, sau lenh nay, S2 thuong xuyen cao (1)
while(1) { // Loop forever
      if ((P1IN & S2) == 0)
       P1OUT |= LED1; // Yes: bat  LED1
      else
      P1OUT &= ~ LED1; // No: tắt LED1, LED2 không thay  doi
            }
}
```

Hãy viết CT bật tắt LED2 bằng công tắc S2. LED1 không thay đổi

Chớp đèn bằng hàm Delay()

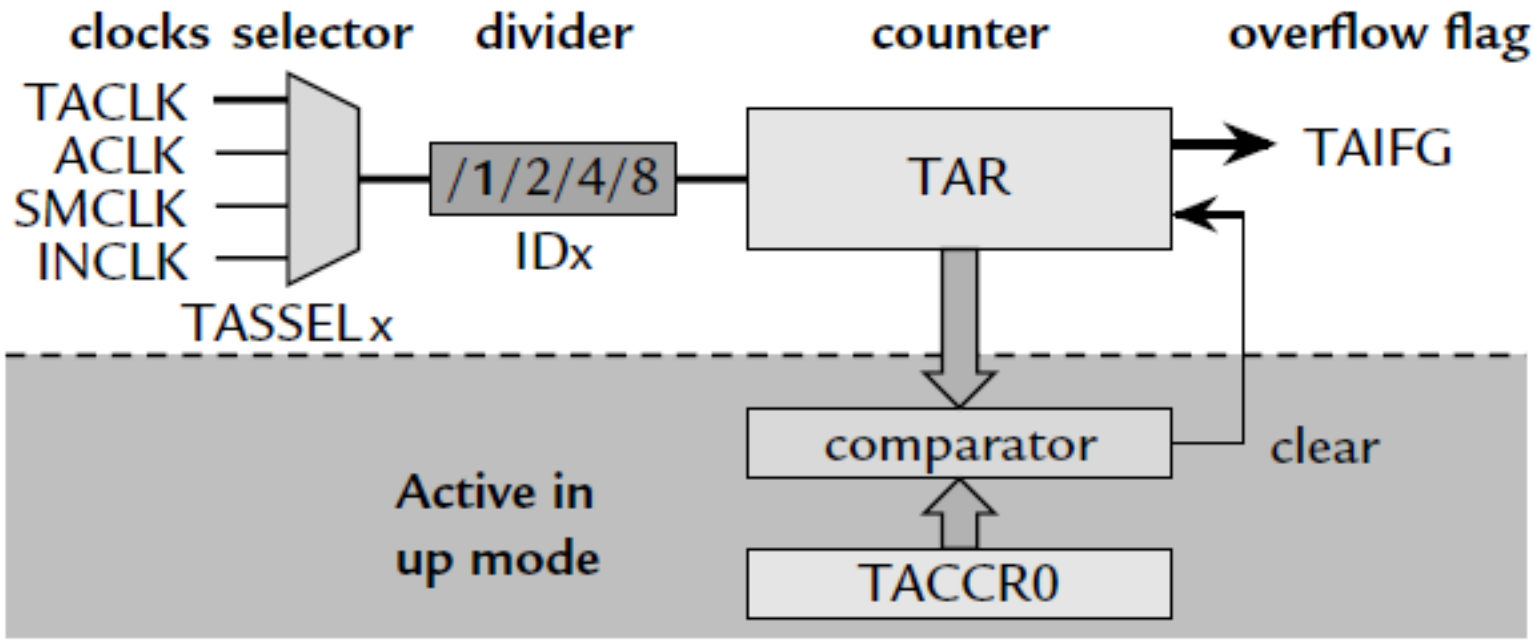
- Chương trình tiếp theo sẽ chớp đèn LED1 (sáng/tắt) với chu kỳ 1 giây. Như vậy thời gian sáng/tắt là 0,5 s

```
while(1){ // Loop forever
for (LoopCtr = 0; LoopCtr < DELAYLOOPS; ++ LoopCtr) {
} // Empty delay loop
P1OUT ^= LED1|LED2; // Toggle LEDs
}
}
```

Tự động chớp đèn: Sử dụng timerA

- Sử dụng vòng lặp là giải pháp đơn giản nhưng không chính xác: khi xảy ra một sự kiện ngắt, vòng lặp sẽ bị kéo dài.
- Đoạn chương trình sau sẽ bật tắt đèn theo timer.
- Timer sẽ được trình bày ở phần sau

TimerA


clocks selector
divider
counter
overflow flag
TACLK
ACLK
SMCLK
INCLK
TASSELx
/1/2/4/8
IDx
TAR
TAIFG
comparator
clear
Active in
up mode
TACCR0

```
#include "msp430g2553.h" //2542
#define S2 BIT3
void main(void) {
WDTCTL = WDTPW |WDTHOLD;
P1DIR = BIT0|BIT6;
P1OUT = 0x00;
  CCTL0 = CCIE;                          // CCR0 interrupt enabled
  CCR0 = 30000;
  TACTL = TASSEL_2 + MC_2 + ID_2;                // SMCLK, upmode
  _BIS_SR( GIE);                // Enter LPM0 w/ interrupt
 while(1){}
}
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
P1OUT ^=BIT0 | BIT6;
CCR0 += 30000;
}
```

Quizzes

- Các phương pháp chớp một đèn LED?

Chương tiếp theo

- ***Hàm, ngắt và các chế độ tiết kiệm năng lượng***

Bài 4: Hàm, ngắt và chế độ tiết kiệm năng lượng

1. **Hàm và các bước thực thi khi gọi hàm**
2. **Ngắt và chương trình phục vụ ngắt**
3. **Các bước thực thi khi xảy ra một ngắt**
4. **Các chế độ tiết kiệm năng lượng**

Hàm

- Hàm là cách viết chương trình thành các mô đun nhỏ. Việc tạo chương trình từ các mô đun khiến chương trình trở nên rõ ràng, dễ viết, dễ kiểm thử và có thể dùng nhiều lần
- Một khi hàm được viết, có thể đóng gói thành thư viện để sử dụng khi cần (thư viện động)
- Các chương trình nhúng viết bằng ngôn ngữ c thường có các hàm dài không quá 30 dòng lệnh

Các bước thực thi khi gọi hàm

- Các thông tin của chương trình đang thực thi được cất vào ngăn xếp
- Địa chỉ của chương trình mới được lưu vào PC. Chương trình mới được thực hiện cho tới khi gặp lệnh return.
- Các thông tin của chương trình cũ được nạp lại từ ngăn xếp. Địa chỉ lệnh chương trình cũ cần tiếp tục thực hiện được nạp vào PC và CPU sẽ tiếp tục chương trình cũ

Ngắt

- Ngắt là cơ chế dừng một chương trình đang chạy để thực hiện một chương trình khác khi xảy ra một sự kiện do phần cứng gây ra.
- Ngắt được dùng khi
  - Một sự kiện khẩn cấp cần được đáp ứng tức thời.
  - Các sự kiện rật chậm.
  - Chuyển CPU từ chế độ ngủ sang tích cực.
  - Gọi hệ điều hành

Chương trình phục vụ ngắt

- Đoạn chương trình được gọi khi xảy ra một ngắt gọi là **Chương trình phục vụ ngắt (** *interrupt service routine* - ISR).
- Chương trình xử lý ngắt giống một hàm, ngoại trừ: Nó được gọi bất kỳ khi nào xảy ra sự kiện gây ra ngắt, và sự kiện này thường không biết trước thời điểm.

Cờ ngắt

- Mỗi ngắt có một cờ ngắt riêng, để chỉ thị rằng đã có sự kiện cần xử lý gấp. Chẳng hạn Timer A có cờ ngắt TAIFG Cờ TAIFG sẽ tự động bật lên khi TAR về 0. Khi cờ ngắt bật lên, chương trình xử lý ngắt tương ứng sẽ được gọi, trừ trường hợp bị che.
- Chương trình xử lý một ngắt chỉ được thực thi khi ngắt này không bị che bởi bit GIE – cho phép ngắt toàn cục. Chương trình chính phải bật bít GIE (thuộc thanh ghi SR) để cơ chế ngắt được phép thực hiện
-

Vector ngắt

- Vector ngắt là vùng nhớ cao trong bộ nhớ MSP430. Vùng này chứa các địa chỉ của các ISR. Khi GIE cao và một cờ ngắt bật lên, địa chỉ của ISR tương ứng sẽ được nạp từ IV vào PC.
- Mỗi ngắt có một thứ tự ưu tiên. Nếu 2 ngắt xảy ra đồng thời, thì ngắt có độ ưu tiên cao hơn sẽ được thực thi. Thự tự ưu tiên cao cao thì vị trí ISR trong IV càng cao.

Các bước thực thi khi một ISR được gọi

1. Nếu CPU đang thực hiện một lệnh – Lệnh đó cần được hoàn tất. Nếu CPU đang ngủ, xung MCLK được cấp để đưa CPU về chế độ tích cực.
2. Cất PC vào ngăn xếp.
3. Cất SR vào ngăn xếp.
4. Chọn IRS có độ ưu tiên cao nhất
5. Xóa cờ ngắt của ISR được chọn.
6. Xóa SR.
7. Đặt địa chỉ của ISR từ IV vào PC.

Độ chậm

- Độ chậm là thời gian từ lúc xảy ra sự kiên tới khi chương trình xử lý ngắt được bắt đầu.
- Trong trường hợp CPU đang thực hiện dở 1 lệnh, thì CPU sẽ cần một nhịp hoàn tất và sáu nhịp thực hiện sáu bước còn lại trước khi bắt đầu ISR.

Khai báo ISR

- **Interrupt Service Routines in C**

#pragma vector = TIMERA0_VECTOR
__interrupt void TA0_ISR (**void**)

```
void main (void)
{
WDTCTL = WDTPW|WDTHOLD; // Stop watchdog timer
P2OUT = ~LED1; // Preload LED1 on , LED2 off
P2DIR = LED1|LED2; // Set pins with LED1 ,2 to output
TACCR0 = 49999; // Upper limit of count for TAR
TACCTL0 = CCIE; // Enable interrupts on Compare 0
TACTL = MC_1|ID_3|TASSEL_2|TACLR; // Set up and start Timer A
// "Up to CCR0" mode , divide clock by 8, clock from SMCLK , clear timer
__enable _interrupt (); // Enable interrupts (intrinsic)
for (;;) { // Loop forever doing nothing
} // Interrupts do the work
}
// -------------------------------------------------------------------
// Interrupt service routine for Timer A channel 0
#pragma vector = TIMERA0_VECTOR
__interrupt void TA0_ISR (void)
{
P2OUT ^= LED1|LED2; // Toggle LEDs
}
```

Ngắt không che

- Có một số ngắt mà cờ GIE không che được là
  - Lỗi bộ tạo dao động OFIFG.
  - Tranh chấp bộ nhớ ACCVIFG.
  - Chân RST bị đưa xuống thấp

Chế độ công suất thấp

- **Có 1 chế độ tích cực và 5 chế độ công suất thấp**
- **Chế độ tích cực:** CPU và mọi xung nhịp, mọi khối đều hoạt động. Dòng tiêu thụ $I \approx 300$uA. Có thể giảm dòng tiêu thụ nếu giảm áp nguồn nuôi xuống 1,8V, tần số DCO =1Mhz, dòng tiêu thụ còn khoảng $I \approx 200$uA.
- **LPM0:** CPU và MCLK bị khóa. SMCLK và ACLK hoạt động, dòng tiêu thụ $I \approx 85$uA.
- **LPM3:** CPU, MCLK, SMCLK, và DCO bị khóa chỉ còng ACLK hoạt động, $I \approx 1$uA.
- **LPM4:** CPU và mọi xung nhịp bị khóa. CPU sẽ chỉ tỉnh lại khi có ngắt bên ngoài. $I \approx 0.1$uA.
- Ví dụ lập trình C

```
_lowpower_mode_3 ();
```

Quizzes

Vì sao Ngắt lại quan trọng với HTN?

Cờ ngắt để làm gì?

Bít GIE là gì?

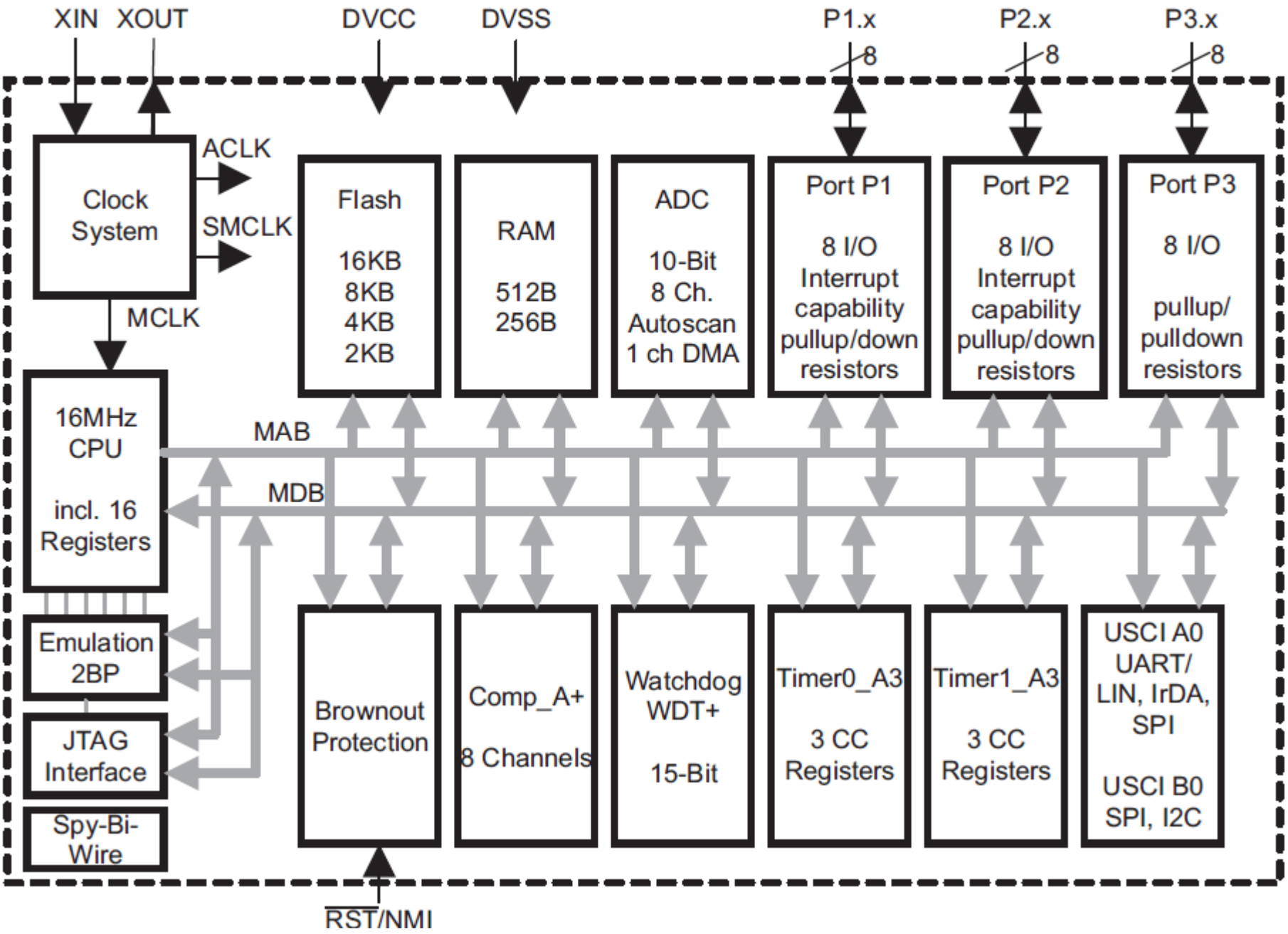
Bài sau?

- ***Cổng nhập xuất số***

Bài 5: Cổng nhập xuất số

1. Cấu trúc cổng nhập xuất số (IO Port)
2. Các thanh ghi của IO
3. Chống dội
4. Ma trận bàn phím
5. Lái LED và LED 7 đoạn
6. LCD

Functional Block Diagram, MSP430G2x53

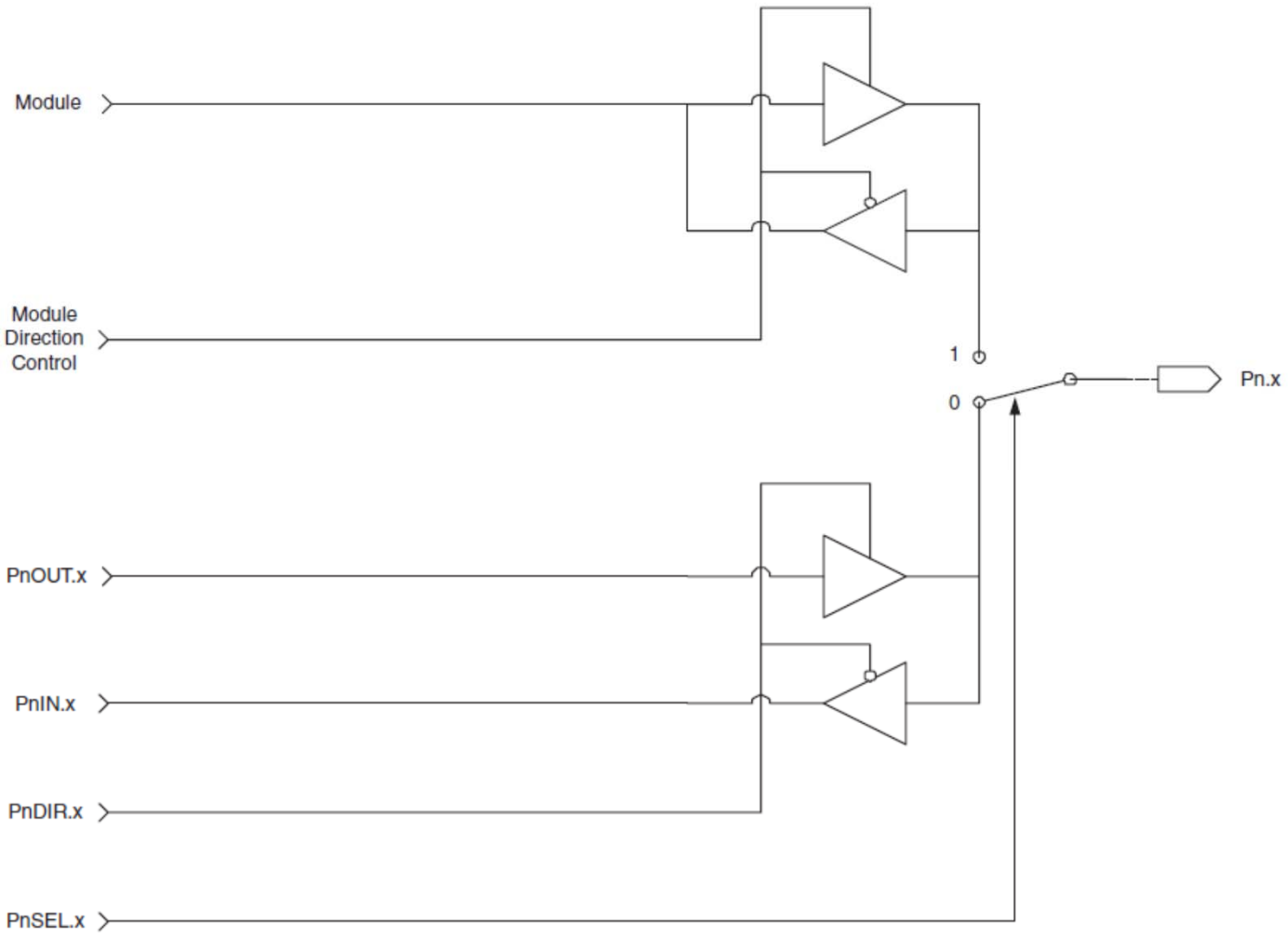


Cổng nhập xuất số

- Nhập xuất thông dụng nhất là các tín hiệu số với 2 giá trị 0/1(0/3,3,V)
- Chương trình ví dụ đã sử dụng PORT 1 để lái LED
- .

Cổng nhập xuất số

- Các chíp MSP430 có thể có từ 10-80 chân IO số.
- Mặc dù là chân nhập xuất số, các chân còn được dùng cho các khối khác như Timer, ADC… Khi tái khởi động các chân đều là IO số
- Chẳng hạn chân P1.0 còn được dùng làm TACLK…


Module
Module
Direction
Control
1
0
Pn.x
PnOUT.x
PnIN.x
PnDIR.x
PnSEL.x

Các thanh ghi của P1

- **Các Port đều có các thanh ghi của riêng mình**
- **Port P1 input, P1IN:** nhận dữ liệu logic (0/1) từ các chân của P1 (8 bít)
- **Port P1 output, P1OUT:** Ghi dữ liệu ra các chân của P1

Các thanh ghi của P1

•**Port P1 direction, P1DIR:** xác định hướng truyền dữ liệu. Nếu bít i =0, chân P1.i sẽ là INPUT, nếu bít i=1, chân P1.i sẽ là OUTPUT . Khi khởi động các bít P1DIR có giá trị 0

•**Port P1 resistor enable, P1REN:** Bật một bít của thanh ghi này lên 1 sẽ kich hoạt điện trở kéo lên hoặc kéo xuống tại chân tương ứng.

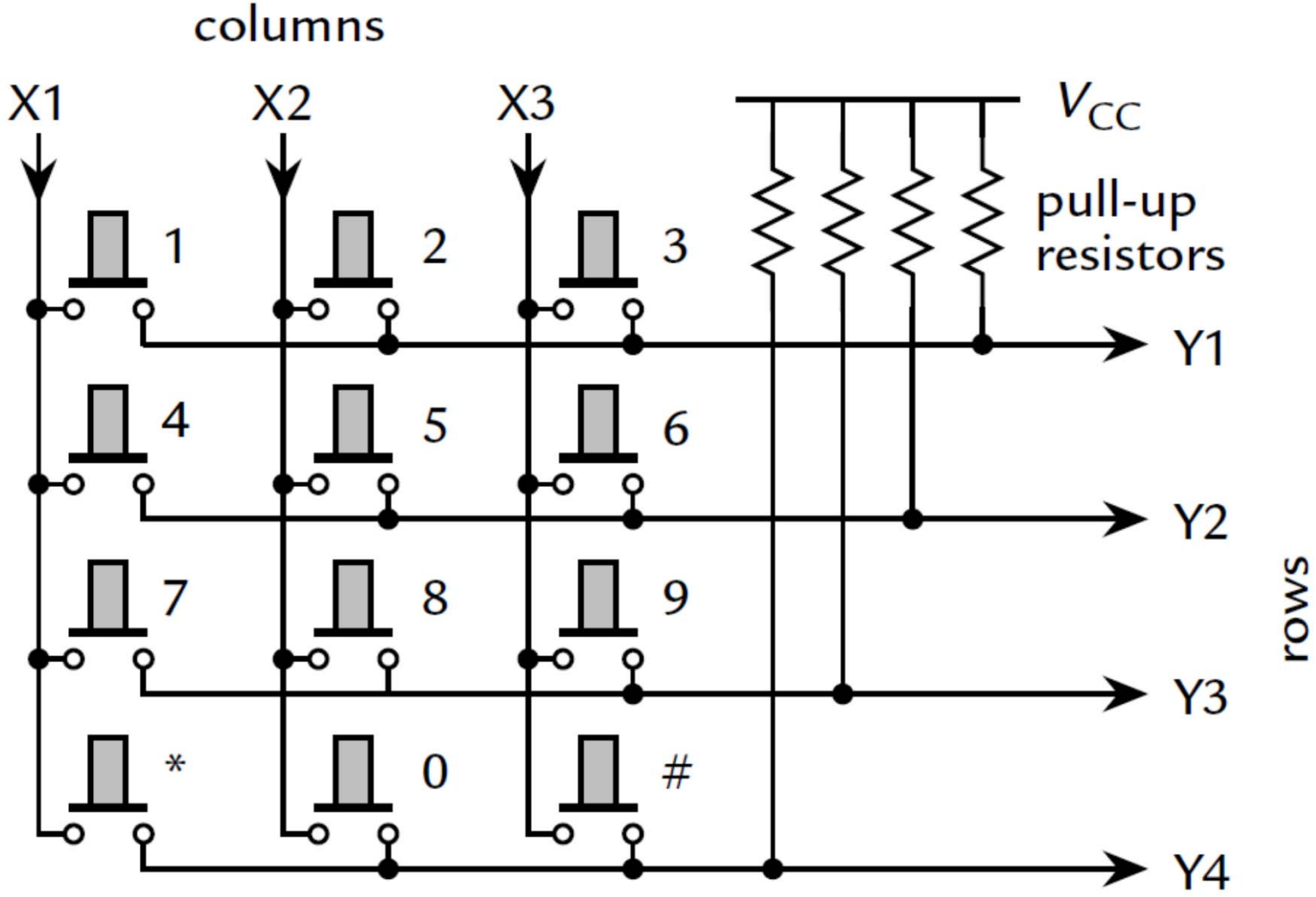
•**Port P1 selection, P1SEL:** Chọn chân tương ứng là chân nhập xuất số (0 – giá thị mặc nhiên khi khởi động) hoặc có chức năng khác (1).

Digital Input and Output

- **Port P1 interrupt enable, P1IE:** Cho phép ngắt trên chân tương ứng nếu bật lên 1, cấm ngắt nếu bít tương ứng là 0
- **Port P1 interrupt edge select, P1IES:** chọn cạnh lên để tạo ngắt nếu bít tương ứng là 0, hoặc cạnh xuống nếu là 1. Thanh ghi này chỉ có hiệu lực nếu thanh ghi P1IE đã bật
- **Port P1 interrupt flag, P1IFG:** Là thanh ghi chứa các cờ ngắt . Khi một chân được bật cho phép ngắt và có sự thay đổi tín hiệu đúng như thanh ghi P1IES đặt thì cò ngắt tu7ong ứng chân này bật lên. Nêu GIE đã bật thì ISR sẽ được thực thi

Quét ma trận bàn phím

- Khi số nút nhấn khá lớn, số chân chíp dùng điều kiển các công tắc sẽ lớn.
- Có thể giảm số chân VDK bắn ma trận bàn phím

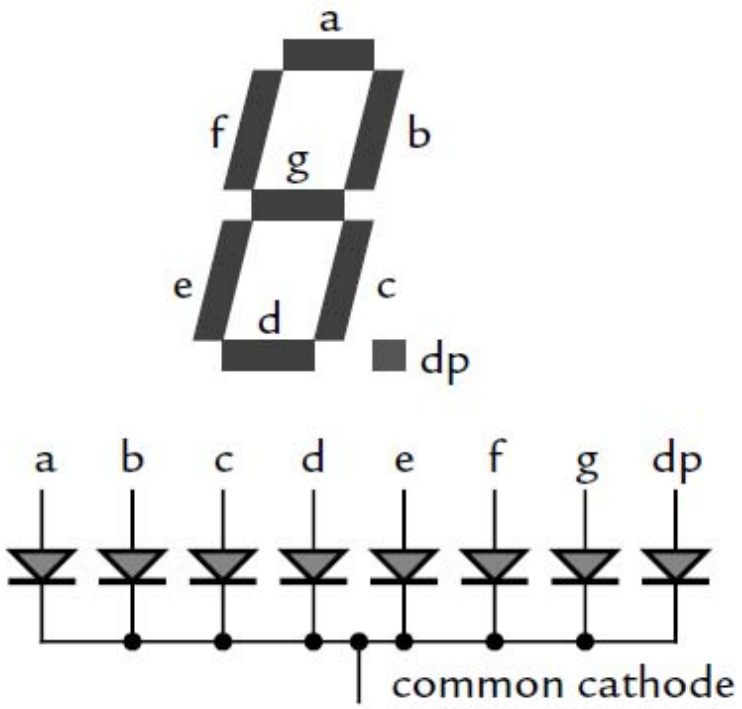

columns
X1
X2
X3
V_{CC}
pull-up
resistors
1
2
3
4
5
6
7
8
9
*
0
#
Y1
Y2
Y3
Y4
rows

Quét bàn phím

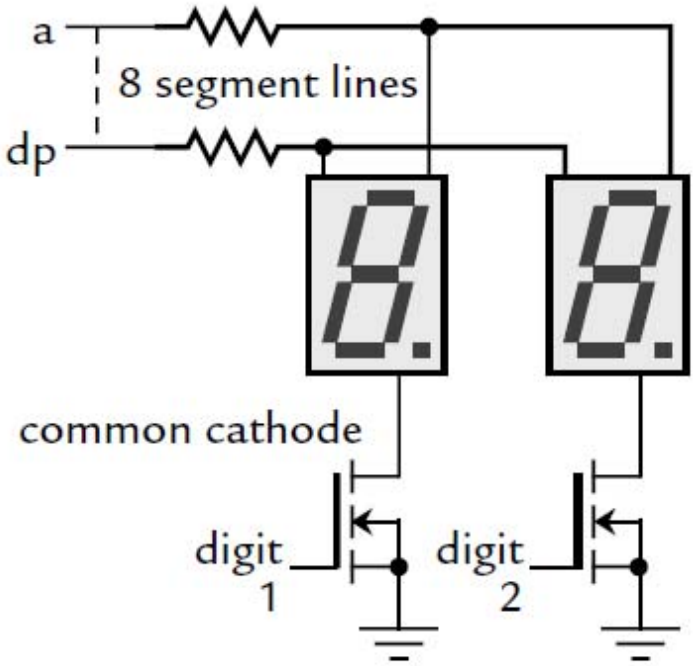
1. Cấp 011 cho X1X2X3 Đọc Y1Y2Y3Y4 để kiểm tra các nút 1, 4, 7, or *. Các nút khác sẽ không làm thay đổi Y1-Y4 vì X2,X3 có giá trị 1
2. Cấp 101 cho X1X2X3 Đọc Y1Y2Y3Y4 để kiểm tra các nút 2, 5, 8, or 0.
3. Cấp 110 cho X1X2X3 Đọc Y1Y2Y3Y4 để kiểm tra các nút 3, 6, 9, or #.

LED 7 đoạn

(a) individual seven-segment LED display



(b) multiplexed pair of displays



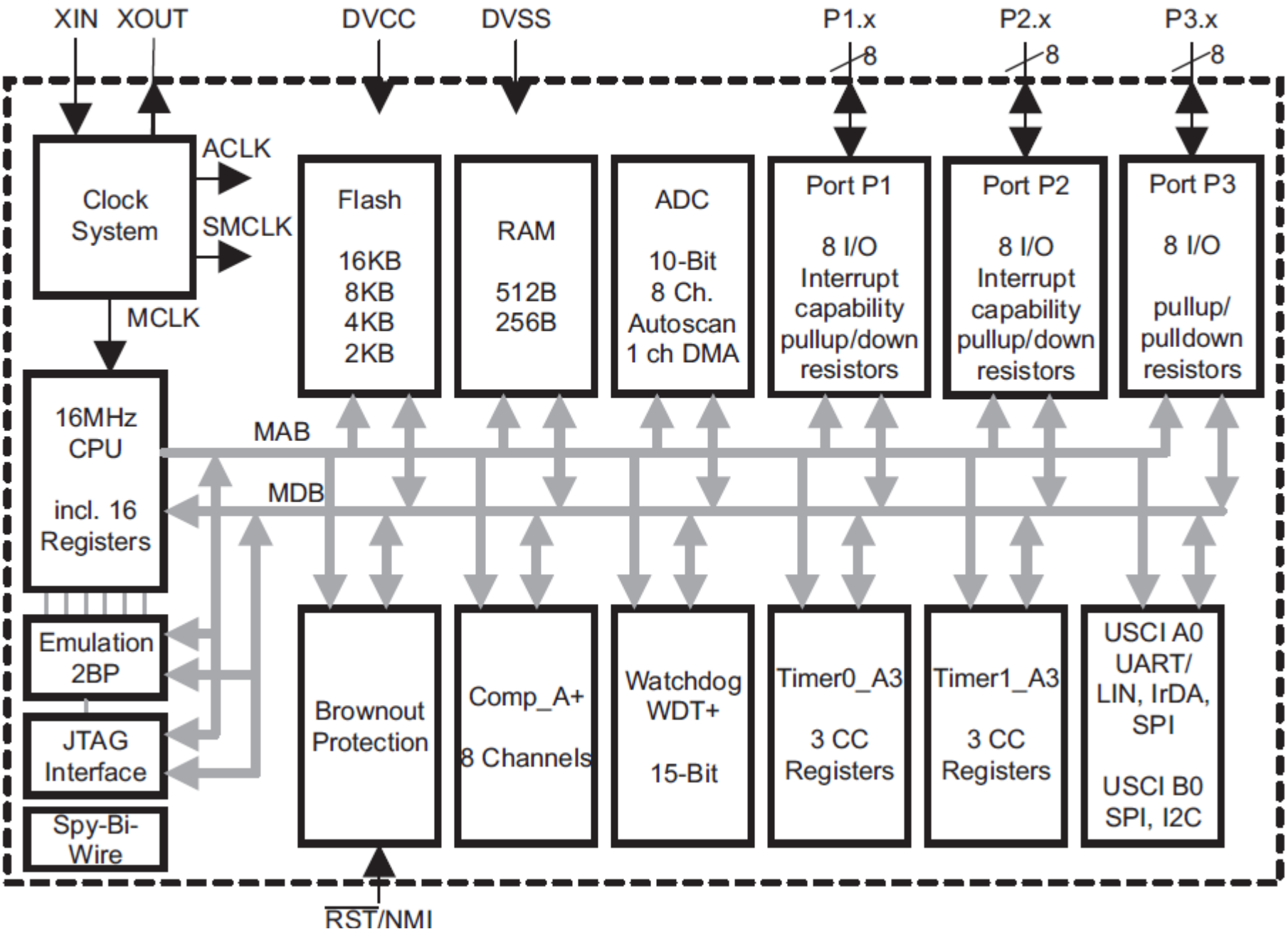
Bài tiếp theo

1. Các loại Timer thường gặp
2. Watchdog Timer
3. Timer A

Bài 6: Bộ định thời

1. Các loại timer
2. Watchdog timer
3. Timer A

Functional Block Diagram, MSP430G2x53



Các loại Bộ định thời

Watchdog timer: Đồng hồ canh gác : có trong mọi thiết bị dòng MSP430. Chức năng chính là chống lỗi chương trinh nhưng có thể sử dụng như một bộ đếm thời gian chính xác

.

Timer_A: Có trong mọi thiết bị. Loại Timer A có 3 kênh và là thiết bị định thời đơn giản nhất. Timer Acó thể đếm thời gian, đo tần số hoặc đếm các sự kiện lặp

Timer_B: Chỉ có trong một số chíp lớn. Đây là cấu trúc mở rộng của Timer A với số kênh lên đến 7 , được sử dụng để tạo các xung PWM điều khiển động cơ

Timer Basic1: Chỉ có trong một số chíp MSP430F4XX.

Real time –clock: Chỉ có trong một số chíp MSP430 lớn

Watchdog Timer (WDT)

- Chức năng chính: chống lỗi chương trình như lặp quẩn, treo VĐK…
- WDT có một thanh ghi đếm lên WDTCNT và khi đạt giới hạn trên (65535) nó sẽ khởi động lại VĐK.
- Chương trình được viết phải định kỳ xóa thanh ghi đếm lên của WDT trước khi đạt giới hạn. Nếu không hệ thống sẽ bị reset.
- Tuy nhiên hoạt động của WDT được cấu hình bởi thanh ghi điều khiển 16-bit WDTCTL. Bật bít WDTHOLT sẽ ngừng hoạt động của WDT

7	6	5	4	3	2	1	0
WDT-HOLD	WDT-NMIES	WDTNMI	WDT-TMSEL	WDT-CNTCL	WDTSSEL	WDTISx	
rw-(0)	rw-(0)	rw-(0)	rw-(0)	r0(w)	rw-(0)	rw-(0)	rw-(0)

Figure 8.1: The lower byte of the watchdog timer control register WDTCTL.

WDT luôn sử dụng xung nhịp DCO 1MHz. Khi hoạt động, bít WDTCNTCL bật lên 1 sẽ xóa thanh ghi WDTCNT

```
// Watchdog config: active , ACLK /32768 -> 1s interval; clear counter
#define WDTCONFIG (WDTCNTCL|WDTSSEL)
// Include settings for _RST/NMI pin here as well
// -------------------------------------------------------------------
void main (void)
{
WDTCTL = WDTPW | WDTCONFIG; // Configure and clear watchdog
P2DIR = BIT3 | BIT4; // Set pins with LEDs to output
P2OUT = BIT3 | BIT4; // LEDs off (active low)
for (;;) { // Loop forever
LED2 = ~IFG1_bit.WDTIFG; // LED2 shows state of WDTIFG
if (B1 == 1) { // Button up
LED1 = 1; // LED1 off
} else { // Button down
WDTCTL = WDTPW | WDTCONFIG; // Feed/pet/kick/clear watchdog
LED1 = 0; // LED1 on
}
}
}
```

Sử dụng WDT như một bộ định thời

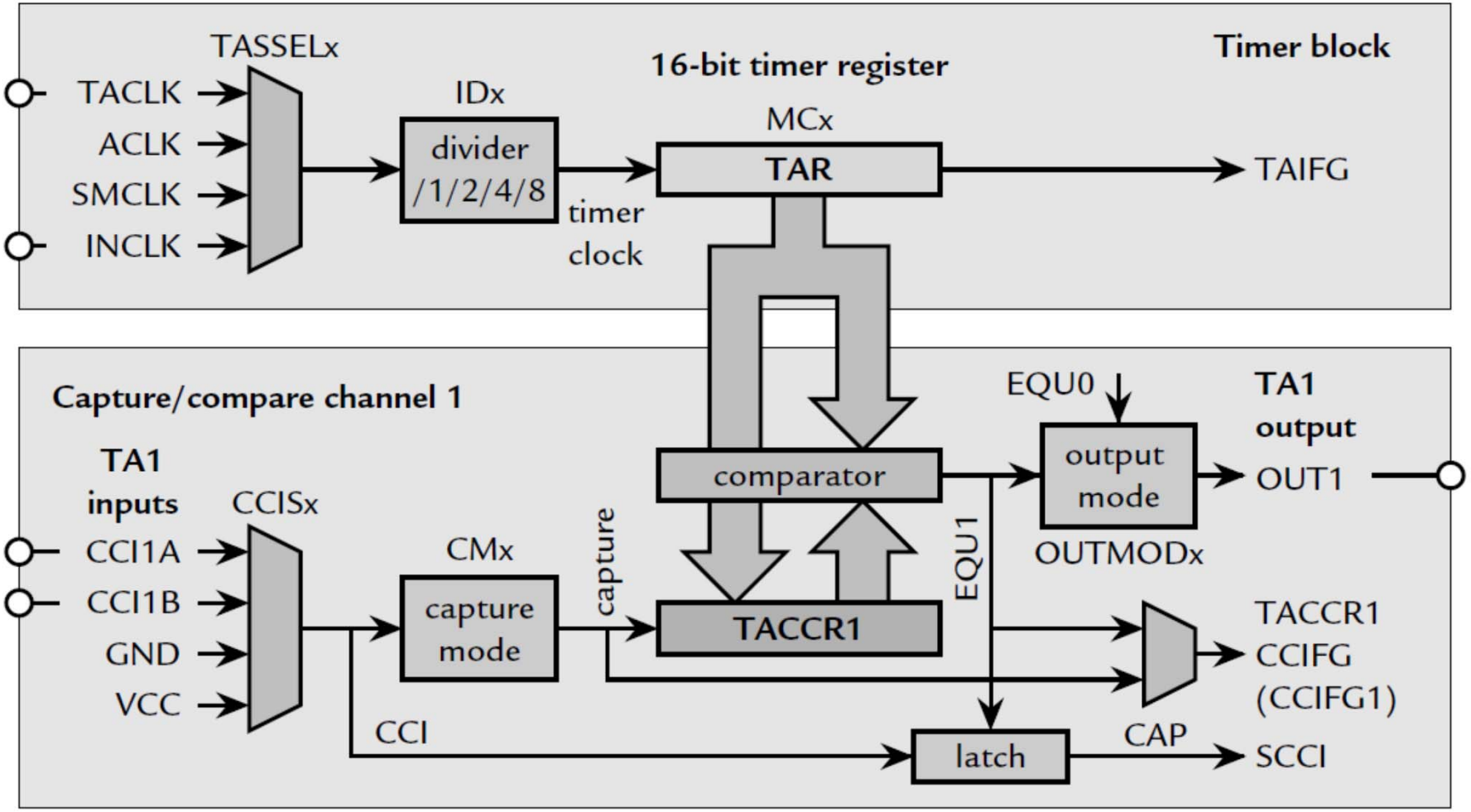
- Nếu không dùng để kiểm tra lỗi chương trình, WDT có thể được sử dụng như một bộ định thời .
- Bật bít WDTTMSEL trong thanh ghi WDTCTL sẽ đặt WDT vào chế độ đếm, tuy nhiên khi đạt giới hạn thì WDTIFG bật lên mà không khởi động lại VĐK
- Việc đọc WDTCNT cho biết thời gian đã xảy một sự kiện

Timer_A

- Là bộ định thời đa năng thông dụng, có mặt trong mọi chíp MSP430. Bộ định thời gồm 2 khối lớn
- **Timer block:** Là khối lõi với thanh ghi TAR 16 bít, khối chọn xung nhịp và khối chia tần làm chậm. Khôi TAR không có tín hiệu ra nhưng có thể bật cờ TAIFG khi TAR về 0
- **Capture/compare channels:** Là khối bắt tín hiệu và so sánh, có 3 khối độc lập với các chỉ số 0,1 và 2. Khối này gồm 0 khối con như sau

- **Khối bắt tín hiệu Capture :** Nhận tín hiệu cần đếm từ bên ngoài
- **Khối so sánh Compare** Gồm thanh ghi TACCR0 (hoặc 1,2) để đếm số tín hiệu hoặc được đặt bằng phần mềm.

- **Khối tạo ngắt : bật cờ** CCIFG khi TAR và TACCR bằng nhau.
- **Khối lấy mẫu Sample** tạo tín hiệu để so sánh
- .


Timer block
TASSELx
TACLK
ACLK
SMCLK
INCLK
IDx
divider
/1/2/4/8
timer
clock
16-bit timer register
MCx
TAR
TAIFG
Capture/compare channel 1
TA1
inputs
CCISx
CCI1A
CCI1B
GND
VCC
CMx
capture
mode
capture
comparator
TACCR1
EQU0
output
mode
OUTMODx
TA1
output
OUT1
EQU1
TACCR1
CCIFG
(CCIFG1)
CCI
latch
CAP
SCCI

Các chế độ định thời

Table 8.1: Resolution and period of Timer_A in the Continuous mode with different clocks and input dividers. Values have been rounded for clarity.

Input clock		Timer clock		Range of timer	
Source	**Frequency**	**Divider**	**Resolution**	**Frequency**	**Period**
SMCLK	16 MHz	1	$\frac{1}{16}$ µs	240 Hz	4 ms
SMCLK	1 MHz	1	1 µs	15 Hz	66 ms
SMCLK	1 MHz	8	8 µs	2 Hz	0.5 s
ACLK	32 KHz	1	31 µs	$\frac{1}{2}$ Hz	2 s
ACLK	32 KHz	8	240 µs	$\frac{1}{16}$ Hz	16 s

Bốn chế độ hoạt động

- **Stop (MC = 0):** Bộ định thời dừng hoạt động. Mọi thanh ghi giữ nguyên giá.
- **Đếm lên (MC=1)** TAR sẽ đếm lên tới TACCR0 (đối với kênh 0) và quay về 0.
- **Liên tục (MC= 2):** Đếm lên 0xFFF rồi quay về 0
- **Đếm lên/xuống(MC= 3):** Thanh ghi TAR đếm lên đến TACCR0 rồi đếm xuống tới 0 và lặp lại
- .

Embedded Systems

Hệ thống nhúng

Ts. Lê Mạnh Hải

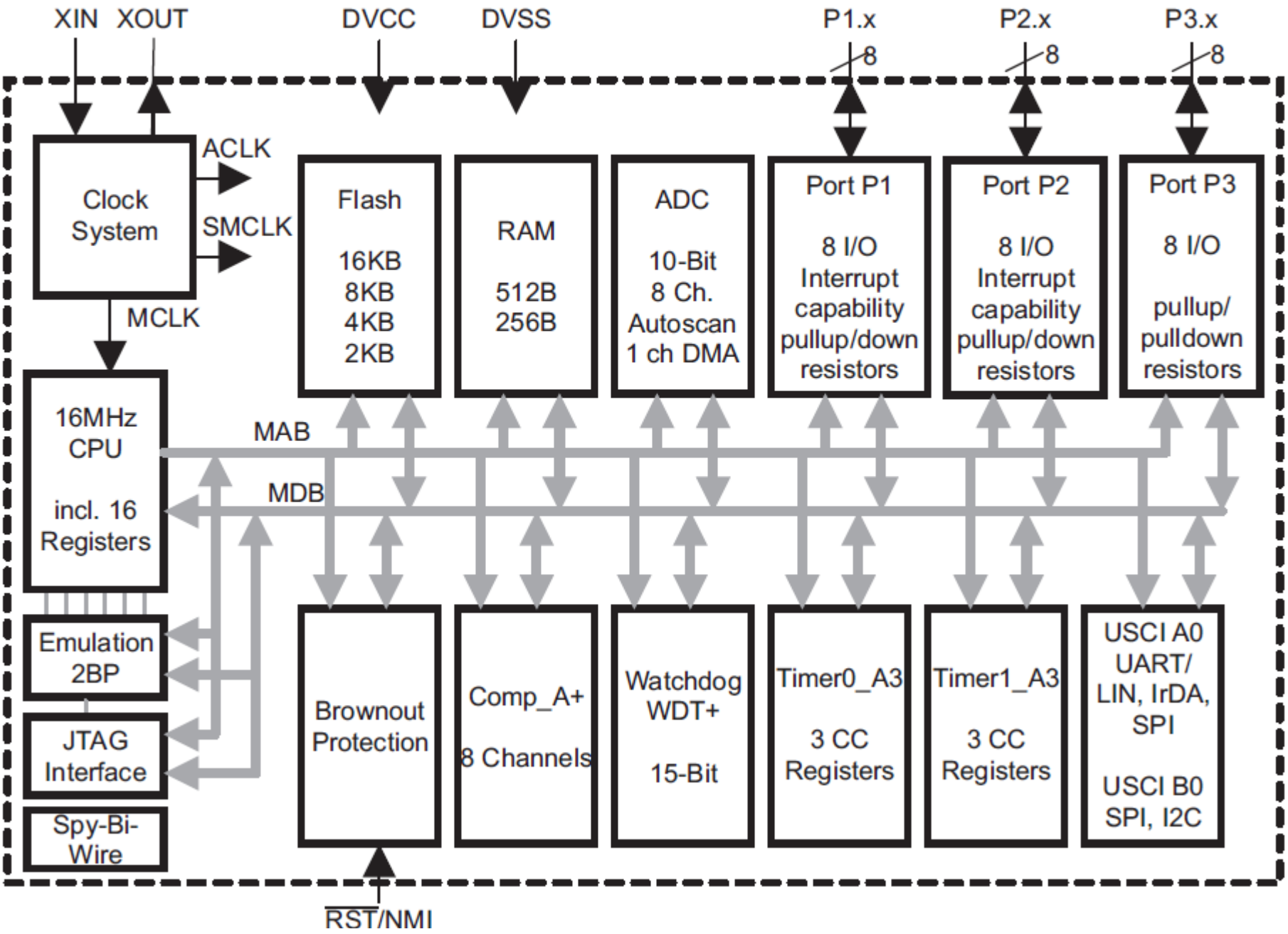
Khoa CNTT,

ĐH Kỹ thuật Công nghệ TP HCM

Chapter 10 : Communication

1. **Communication Peripherals in the MSP430**
2. **Serial Peripheral Interface**
3. **SPI with the USI**
4. **SPI with the USCI**
5. **Inter-integrated Circuit Bus**
6. **A Simple I²C Master with the USCI_B0**
7. **A Simple I²C Slave with the USI on a F2013**

Functional Block Diagram, MSP430G2x53



Communication Peripherals in the MSP430

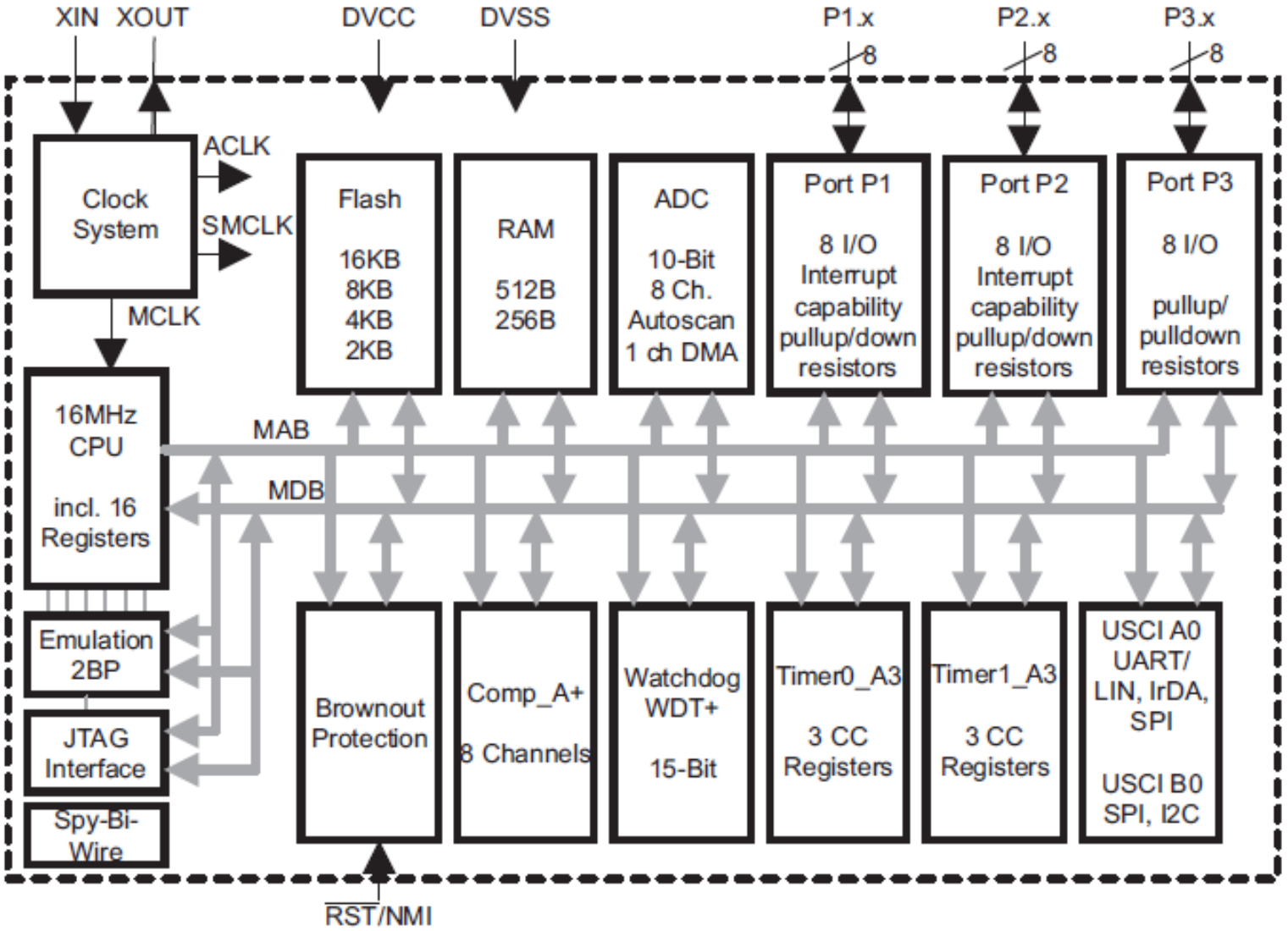
- The universal serial interface (USI) is a lightweight module, which is included in the small F20x2 and F20x3 devices. For a start, it handles only synchronous communication—SPI and I²C.
- **Universal Serial Communication Interface**

larger devices in the MSP430F2xx and MSP430F4xx families contain one or more *universal serial communication interface* (USCI) modules. The hardware handles almost all aspects of the communication, unlike the USI, so the software needs only to provide the data to transmit and store the received data in normal operation. Typically this requires only a couple of small interrupt service routines.

- **Universal Synchronous/Asynchronous Receiver/Transmitter**

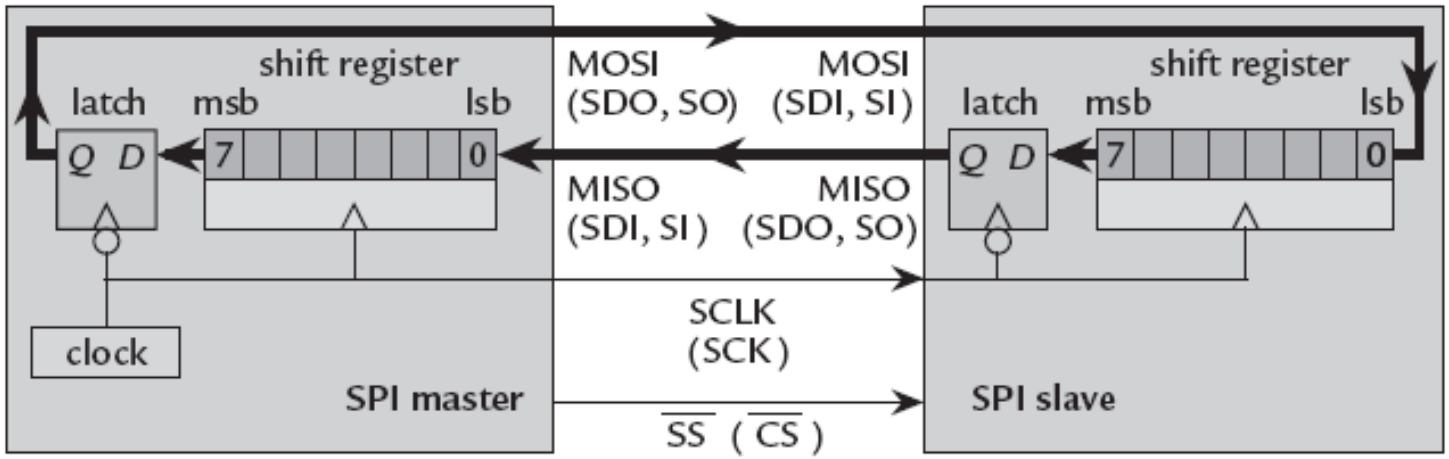
(USART) is an older module, which has been superseded by the USCI.

Functional Block Diagram, MSP430G2x53



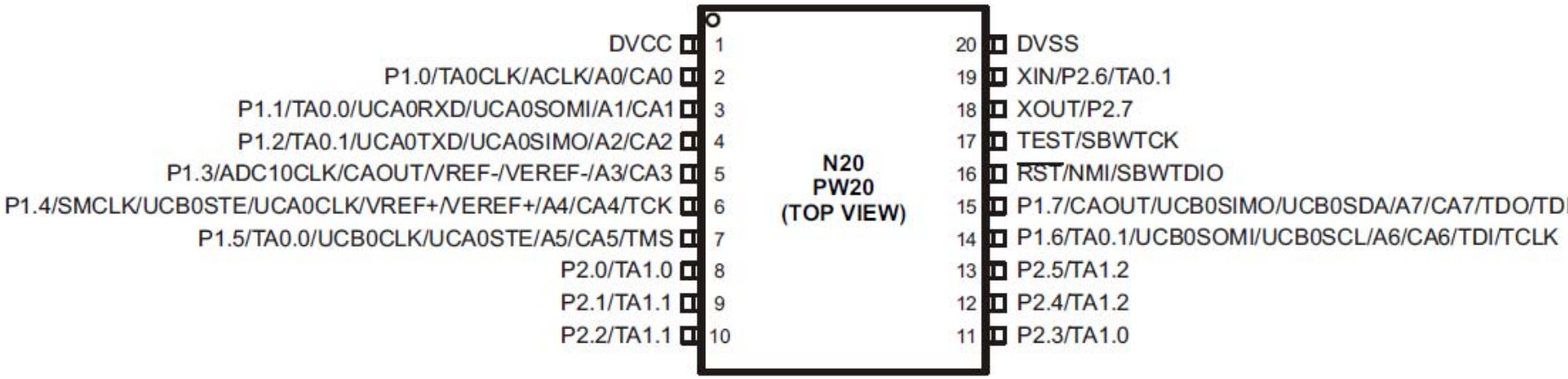
Serial Peripheral Interface

- The serial peripheral interface was introduced by Motorola and is the simplest synchronous communication protocol in general use.
- The only problem is that it is not a fixed standard like I²C. There are plenty of options within "standard" SPI and innumerable variations that go beyond this.



- The concept of SPI is based on two shift registers, one in each device, which are connected to form a loop.
- The registers usually hold 8 bits. Each device places a new bit on its output from the most significant bit (msb) of the shift register when the clock has a negative edge and reads its input into the lsb of the shift register on a positive edge of the clock.
- Thus a bit is transferred in each direction during each clock cycle.
- After eight cycles the contents of the shift registers have been exchanged and the transfer is complete.
- Transmission and reception are clearly inseparable.

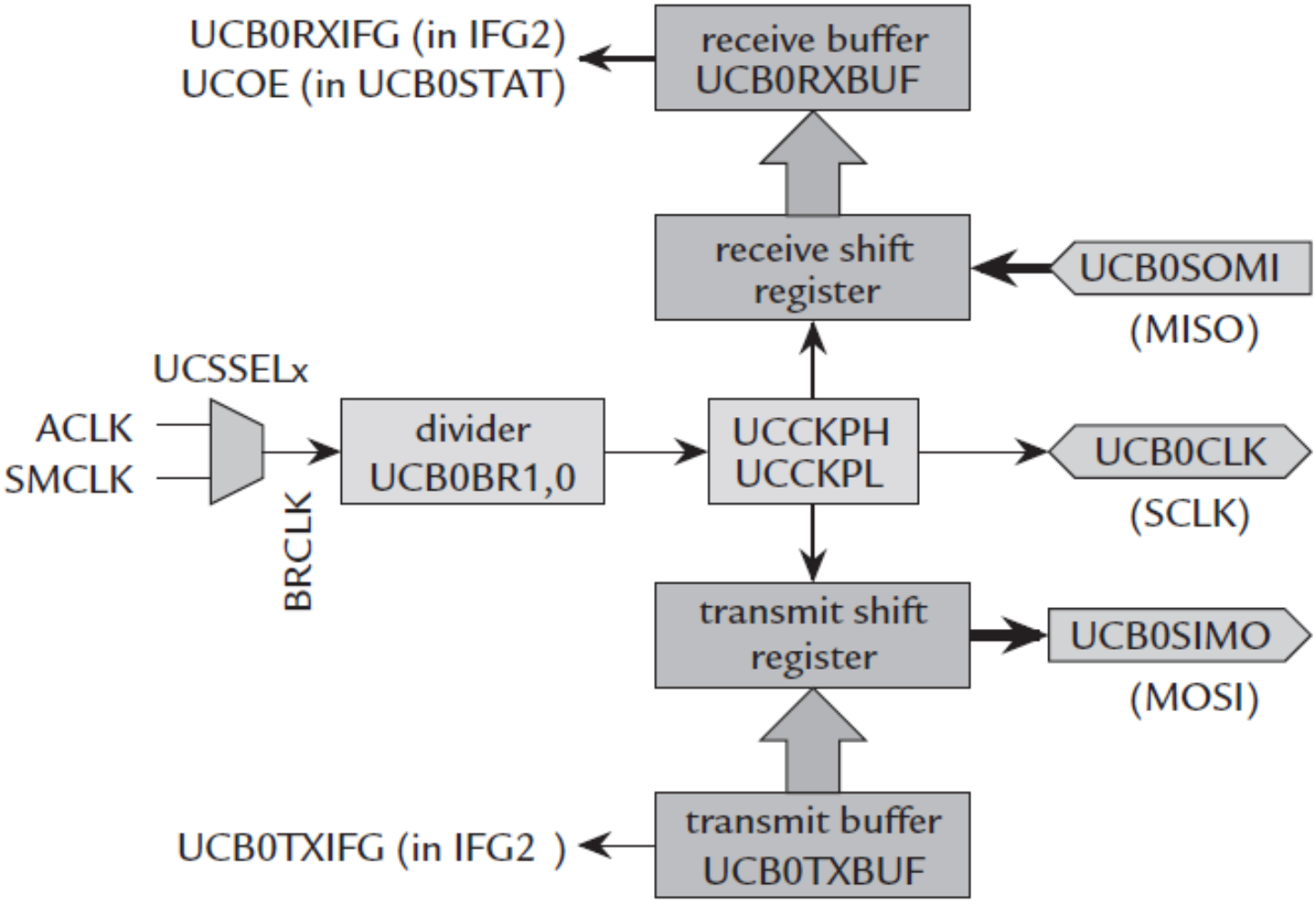
The main pins are labeled SOMI, SIMO, and CLK (2 USCI)



NOTE: ADC10 is available on MSP430G2x53 devices only.

NOTE: The pulldown resistors of port P3 should be enabled by setting P3REN.x = 1.

SPI block in USCI



SPI operation

There are separate shift registers for transmitting and receiving
Moreover, these registers are double-buffered and the user has no direct access to the shift registers themselves.
This means that a byte is moved from the receive shift register to RXBUF as soon as reception is complete, which leaves the shift register ready to accept the next transfer.
Similarly, a byte written to TXBUF remains in its buffer until the previous byte has been transmitted, at which point it is moved to the transmit shift register.
This relaxes considerably the constraints on handling interrupts in the USI, where the shift register must be read and updated rapidly between transfers. Although there are separate registers, reception and transmission are not independent because of the nature of SPI.

Read an examples C code for SPI communication

For Master

msp430g2xx3_uscia0_spi_09.c

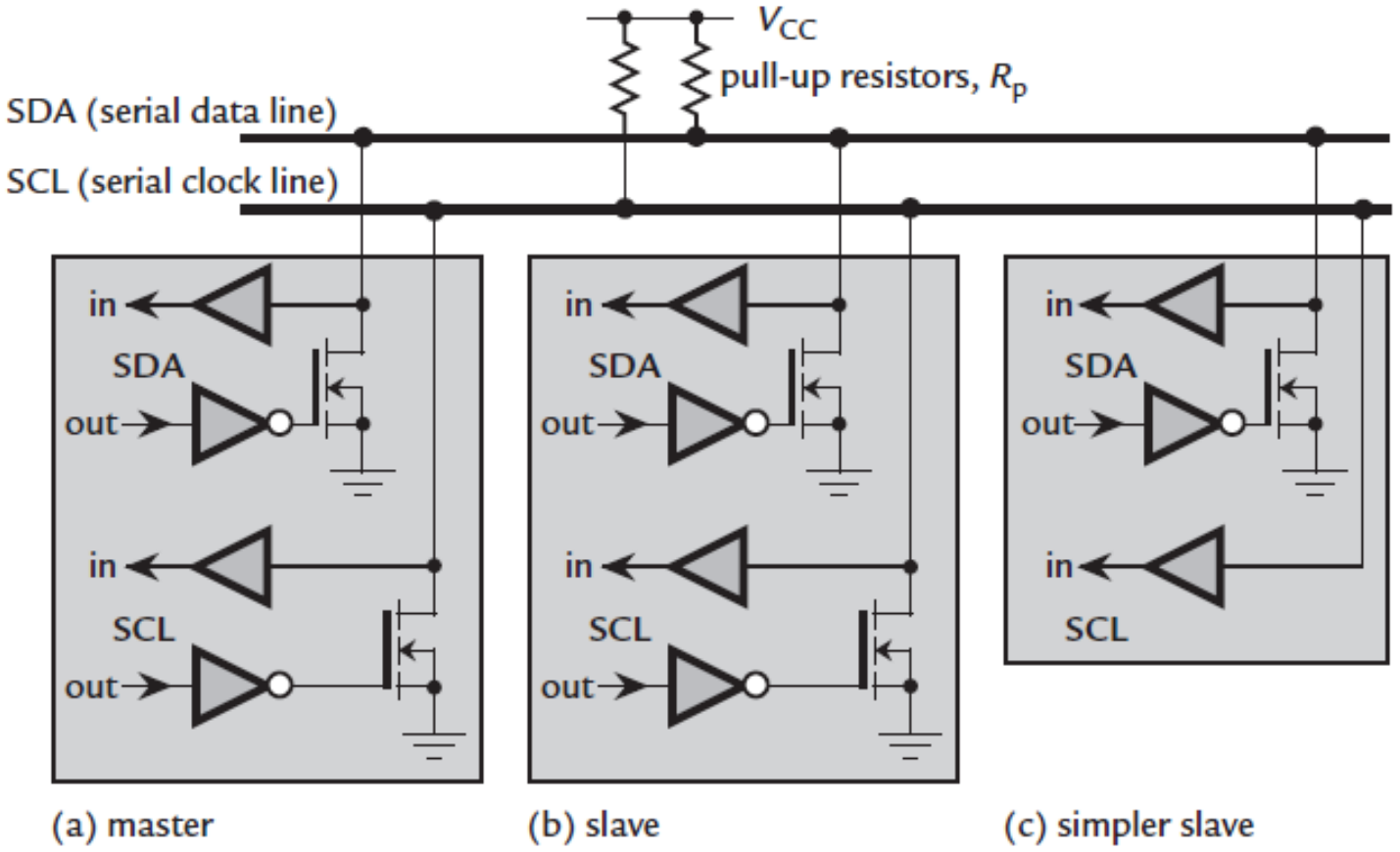
And for Slave

msp430g2xx3_uscia0_spi_10.c

Inter-integrated Circuit Bus

- The I²C bus was introduced by Philips (now NXP) Semiconductors. It was widely adopted and has become even more popular since its patents expired in 2006. It is a true bus, unlike SPI, with a specification and user manual that can be downloaded from NXP.
- Revision 03 of the user manual is document UM10204, dated June 19, 2007. It is clearly written and a lot easier to read than you might expect. The I²C bus uses only two, bidirectional lines:
- Serial data (SDA).
- Serial clock (SCL).

- Structure



- Operation: Read from page 534 -542

Ôn tập Hệ thống nhúng

1. Cấu trúc tổng thể của vi điều khiển
2. Sơ đồ khối của chíp TI MSP430G2553
3. Bộ nhớ MSP430G2553: Phân bổ vị trí bộ nhớ và ý nghĩa từng vùng nhớ
4. Cấu tạo CPU và ý nghĩa các thanh ghi trong CPU
5. Các loại xung nhịp (clock) và các chế độ hoạt động
6. Hàm và các bước thực hiện khi gọi một hàm
7. Khái niệm ngắt và chương trình phục vụ ngắt
8. Các bước thực thi khi thực hiện một ngắt.
9. Các chế độ công suất thấp.
10. Các cổng nhập xuất số (Digital Input and Output).
11. Quét ma trận bàn phím. Chống dội
12. Các loại LCD. Sơ đồ chân kết nối theo chuẩn HD44780
13. Các loại timer. Cấu trúc và hoạt động của WDT
14. Cấu trúc và các chế độ hoạt động của TimerA0
15. Kết nối Serial Peripheral Interface (SPI). Cấu trúc và hoạt động.
16. Kết nối Inter-integrated Circuit Bus (I2C).

Kiểm tra giữa kỳ

- Viết CT hiển thị phút và giây trên Led 7 đoạn. Sử dụng Timer A (7 điểm)
- Cứ 1 mỗi giây đèn LED L4 phải sáng / tắt dấu chấm (LED h) 1 lần